



**Universität  
Zürich**<sup>UZH</sup>

# Evaluation of Dialogue Systems

Thesis

presented to the Faculty of Arts and Social Sciences  
of the University of Zurich  
for the degree of Doctor of Philosophy

by

Jan Milan Deriu

Accepted in the fall semester 2021

on the recommendation of the doctoral committee composed of

Prof. Dr. Martin Volk (main supervisor)

Prof. Dr. Eneko Agirre

Zurich, 2021

## Abstract

We investigate evaluation methods for dialogue systems. We focus on conversational dialogue systems and question-answering dialogue systems since each class poses different challenges. For conversational dialogue systems, we tackle the inefficiency and unreliability of the evaluation process. For question-answering dialogue systems, we tackle the problem of evaluation after deployment. The main contributions are:

- We introduce a novel paradigm for evaluating conversational agents based on the bot-to-bot talk. This new paradigm allows sampling conversations automatically, reducing human involvement in the evaluation process. We apply this paradigm to two scenarios -first, Spot The Bot, an evaluation procedure based on bot-bot talk. Humans read automatically generated dialogues and decide for each interlocutor, whether it is a human or a bot. Based on this feedback, the bots are ranked. We show that the rankings are robust and reproducible. The second scenario is AutoJudge, an automated metric for evaluating conversational dialogue systems. It is trained on automatically generated dialogues that are annotated on the turn level by human judges. We show that AutoJudge achieves good correlation scores with humans and that it can be used as a meta-selection model to select the best answer from different dialogue systems.
- We introduce a novel evaluation procedure for question-answering dialogue systems over databases, which allows evaluation after deployment. That is, the system can be evaluated without the need for a gold-standard reference. The method is based on back-translating the generated SQL query to a synthetic question. Textual semantic similarity is applied to the original user input and the synthetic question to determine if the underlying SQL query is correct.
- We introduce a novel annotation procedure to generate pairs of questions and queries more efficiently. The procedure is based on inverting the process, i.e., we first sample a structured query from a context-free grammar, and then humans write the corresponding question. We show a 4-fold improvement in the time needed to generate data with respect to traditional approaches where experts write both the question and the query.

---

## Zusammenfassung

Wir untersuchen Methoden zur Evaluation von Dialogsystemen. Dabei fokussieren wir uns auf Unterhaltungsdialogsysteme und Frage-Antwort-Dialogsysteme. Jede Klasse von Dialogsystemen bringt verschiedene Herausforderungen mit sich. Wir gehen das Problem mit ineffizienten und unzuverlässigen Evaluationsprozessen bei Unterhaltungsdialogsystemen an. Bei Frage-Antwort-Dialogsystemen untersuchen wir Methoden, um das System während des Einsatzes zu evaluieren. Die Hauptbeiträge dieser Arbeit sind:

- Spot The Bot ist ein Evaluationsprozess, welcher auf automatisch generierten Dialogen zwischen zwei Systemen basiert. Menschen lesen diese Dialoge und entscheiden für jeden Gesprächsteilnehmer, ob es sich um einen Menschen oder ein Dialogsystem handelt. Basierend auf diesen Annotationen erstellen wir Ranglisten für die Systeme, welche robust und zuverlässig sind.
- AutoJudge ist eine automatisierte Evaluationsmethode für Unterhaltungsdialogsysteme. Wir trainieren AutoJudge auf Dialogen, welche auf der Aussageebene annotiert wurden. Wir zeigen, dass AutoJudge gut mit den menschlichen Annotationen korreliert. Zudem eignet es sich als Auswahlmodul, welches aus einer Menge von Antwortkandidaten die beste Antwort auswählt.
- Für Frage-Antwort-Systeme über Datenbanken haben wir eine neue Evaluationsmethode entwickelt. Dies erlaubt, dass das System während des Einsatzes evaluiert werden kann. Dadurch kann das System ohne Goldstandard evaluiert werden. Die Methode basiert darauf, dass die generierte SQL-Abfrage automatisch zurück in eine natürlichsprachige Frage umgewandelt wird. Diese synthetische Frage wird dann mit der Eingabe des Users bezüglich ihrer semantischen Äquivalenz verglichen. So entscheidet es, ob die unterliegende SQL-Abfrage korrekt ist.
- Wir präsentieren einen neuen und effizienteren Datenannotationsprozess zum Generieren von Frage-Antwort Paaren. Der Prozess basiert darauf, den Ablauf zu invertieren. Das bedeutet, dass wir erst eine Abfrage zufällig aus einer kontextfreien Grammatik erzeugen und Annotatoren die entsprechende Frage stellen lassen. Im Vergleich zu der herkömmlichen Prozedur, bei der Experten Fragen und Antworten schreiben, verzeichnen wir eine Vervierfachung der Annotationsgeschwindigkeit.

# Acknowledgement

Research is never performed in a vacuum. To perform the research one desires, many things need to be in place to enable this endeavor. Luckily this was the case for me, and there are many people that I need to thank for this.

First and foremost, I thank Prof. Dr. Mark Cieliebak, who made all of this possible. He made sure that I could dedicate most of my time to my Ph.D. project. He enabled me to pursue the research that I was interested in and supported me in any way. For his great mentorship, I am entirely grateful.

I thank Prof. Dr. Eneko Agirre for initiating and coordinating the LIHLITH project, which inspired many exciting research collaborations. Furthermore, I thank him for the valuable feedback that he provided to improve this thesis.

I thank Prof. Dr. Martin Volk for accepting me as a Ph.D. student and his valuable feedback to make this thesis as strong as possible.

I thank Dr. Don Tuggener and Pius von Däniken for the great and fun collaboration on the Spot The Bot paper. Especially for the implementation of the Survival Analysis and the IAA analysis. I also thank Pius von Däniken for the implementation of the back-translation.

I thank Katsiaryna Mlynchyk for her help in creating the statistics of the OTTA corpus and her invaluable help in coordinating the whole administrative part of the annotation procedure.

I thank Prof. Dr. Kurt Stockinger for being a great brainstorming partner, especially during the conception of the OTTA corpus.

I thank the Institute of Applied Information Technology at the Zurich University of Applied Sciences, especially Remo Maurer, for providing an excellent infrastructure. This allowed me to run many compute-intensive experiments and deploy all the annotation tools with ease.

Finally, I thank the Ph.D. Network in Data Science, who enabled me to pursue my Ph.D. at the Zurich University of Applied Sciences and the University of Zurich.

---

This work has been primarily funded by the LIHLITH project supported by the EU ERA-Net CHIST-ERA and the Swiss National Science Foundation under the grant 20CH21\_174237.

# Contents

<b>Abstract</b>	<b>i</b>
<b>Acknowledgement</b>	<b>iii</b>
<b>Contents</b>	<b>v</b>
<b>List of Figures</b>	<b>ix</b>
<b>List of Tables</b>	<b>x</b>
<b>List of Acronyms</b>	<b>xi</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Problem Statement for this Thesis . . . . .	5
1.1.1 Increased Efficiency of Evaluation . . . . .	5
1.1.2 Unreferenced Evaluation after Deployment . . . . .	6
1.2 Thesis Contributions . . . . .	6
1.3 Outline . . . . .	8
<b>2 Dialogue Systems</b>	<b>9</b>
2.1 General Overview . . . . .	9
2.1.1 Classification . . . . .	9
2.1.2 General Dialogue Structure . . . . .	11
2.2 Conversational Dialogue Systems . . . . .	11
2.2.1 Neural Conversational Dialogue Systems . . . . .	12
2.2.2 Generative Approach . . . . .	13
2.2.2.1 Encoder-Decoder Based Approaches . . . . .	13
2.2.2.2 Language Model Approaches . . . . .	17
2.2.3 Retrieval Approach . . . . .	17
2.2.4 Mixed Approach . . . . .	18
2.3 Evaluation of Conversational Dialogue Systems . . . . .	18
2.3.1 Human Based Evaluation . . . . .	19
2.3.2 Automated Evaluation Methods . . . . .	20

2.3.2.1	Human Likeness . . . . .	21
2.3.2.2	Appropriateness . . . . .	21
2.4	Natural Language Interfaces to Databases . . . . .	23
2.4.1	Grammar Net . . . . .	24
2.5	Evaluation of Natural Language Interfaces to Databases . . . . .	26
2.5.1	Metrics . . . . .	26
2.5.2	Train-Test Split . . . . .	26
2.5.3	Evaluation after Deployment . . . . .	27
2.6	Corpora . . . . .	28
2.6.1	Conversational Dialogue Systems . . . . .	29
2.6.1.1	Persona-Chat . . . . .	29
2.6.1.2	Dailydialog . . . . .	30
2.6.1.3	Empathetic Dialogues . . . . .	31
2.6.1.4	Twitter Conversation Corpus . . . . .	31
2.6.2	Natural Language Interfaces to Databases . . . . .	32
2.6.2.1	Human Annotated Datasets . . . . .	32
2.6.2.2	Semi-automated Datasets . . . . .	33

### **3 Bot-Bot-Talk: A new Paradigm for the Evaluation of Conversational Dialogue Systems** **36**

3.1	Spot The Bot . . . . .	36
3.1.1	Definition of Spot The Bot . . . . .	37
3.1.2	Output of Spot The Bot . . . . .	39
3.1.2.1	Ranking of Chatbots . . . . .	39
3.1.2.2	Survival Analysis . . . . .	41
3.1.3	Experiments: Application of Spot The Bot . . . . .	42
3.1.3.1	Ranking Results . . . . .	44
3.1.3.2	Survival Analysis Results . . . . .	45
3.1.4	Discussion of Spot The Bots' Advantages . . . . .	48
3.1.4.1	Inter-Annotator Agreement . . . . .	48
3.1.4.2	Reproducibility and Robustness . . . . .	49
3.1.4.3	Time Efficiency . . . . .	50
3.1.4.4	Stability against weak Annotators . . . . .	51
3.1.5	Qualitative Analysis . . . . .	52
3.1.6	Automation of Spot The Bot . . . . .	54
3.2	AutoJudge . . . . .	58
3.2.1	The Process to build AutoJudge . . . . .	58
3.2.1.1	Dialogue Systems . . . . .	60
3.2.1.2	Turn-Level Annotation . . . . .	61

3.2.1.3	Training AutoJudge . . . . .	61
3.2.1.4	Improving Dialogue Systems with AutoJudge . . . . .	62
3.2.1.4.1	Ranking. . . . .	62
3.2.1.4.2	Reinforcement Learning Reward. . . . .	62
3.2.2	Results and Discussion . . . . .	62
3.2.2.1	Data Aggregation . . . . .	63
3.2.2.2	AutoJudge . . . . .	63
3.2.2.3	Answer Selection. . . . .	64
3.2.2.4	Reinforcement Learning. . . . .	65
3.3	Summary . . . . .	68
<b>4</b>	<b>Evaluation for Natural Language Interfaces to Databases</b>	<b>69</b>
4.1	OTTA - A corpus for NLIDB . . . . .	70
4.1.1	Operation Trees . . . . .	70
4.1.2	Inverted Annotation Procedure . . . . .	74
4.1.2.1	Tree Sampling . . . . .	74
4.1.2.2	Annotation Procedure . . . . .	77
4.1.2.2.1	Phase 1 - Question Annotation . . . . .	77
4.1.2.2.2	Phase 2 - Token Assignment . . . . .	80
4.1.3	The OTTA Corpus . . . . .	80
4.1.3.1	Corpus Statistics . . . . .	82
4.1.3.2	Corpus Comparison . . . . .	83
4.1.4	Baseline Model . . . . .	85
4.1.4.1	Model Description . . . . .	87
4.1.4.2	Baseline Results . . . . .	89
4.2	Unreferenced Evaluation based on Back-translation . . . . .	91
4.2.1	Back Translation . . . . .	91
4.2.2	Evaluation Method . . . . .	95
4.2.2.1	Experimental Setup. . . . .	95
4.2.2.2	Results . . . . .	97
4.3	Back-translation and Semantic Textual Similarity for Re-ranking . . .	99
4.3.1	Data Augmentation . . . . .	99
4.3.2	Ranking Hypotheses based on Semantic Textual Similarity . . .	99
4.3.3	Experimental Setup . . . . .	100
4.3.4	Results . . . . .	102
4.4	Summary . . . . .	104
<b>5</b>	<b>Conclusion and Outlook</b>	<b>108</b>
5.1	Conversational Dialogue Systems . . . . .	109



5.2	NLIDB . . . . .	111
5.3	Transfer to another Language . . . . .	112
5.3.1	Conversational Dialogue Systems . . . . .	113
5.3.2	NLIDB . . . . .	113
<b>References</b>		<b>115</b>

# List of Figures

1	HRED - Overview . . . . .	15
2	AST - Example . . . . .	24
3	NLIDB Feedback - Example . . . . .	28
4	Persona-Chat - Example . . . . .	29
5	DailyDialog - Example . . . . .	30
6	EmpatheticDialogues - Example . . . . .	31
7	Spider - Example . . . . .	32
8	ATIS - Example . . . . .	33
9	Overnight - Process . . . . .	34
10	Spot The Bot - Overview . . . . .	38
11	Spot The Bot - Annotation Tool . . . . .	40
12	Spot The Bot - Survival Analysis . . . . .	46
13	Spot The Bot - Ranking Stability . . . . .	49
14	Spot The Bot - Confidence . . . . .	56
15	AutoJudge - Overview . . . . .	59
16	AutoJudge - Reward over time . . . . .	66
17	OTTA - Domain Example . . . . .	70
18	OTTA - Token Assignment . . . . .	72
19	OTTA - Grammar . . . . .	72
20	OTTA - Sampling Example . . . . .	76
21	OTTA - Annotation Process . . . . .	78
22	OTTA - M-to-N Problem . . . . .	79
23	OTTA - Annotation Tool . . . . .	81
24	OTTA - Unreferenced Evaluation Method . . . . .	92
25	OTTA - Back-translation Example . . . . .	93
26	OTTA - Precision Recall Space . . . . .	105
27	OTTA - Improvement . . . . .	106
28	OTTA - Rerank and Pre-train Learning Curves . . . . .	107

# List of Tables

1	Intro - Example . . . . .	4
2	Dialogue System - Classification . . . . .	10
3	Spot The Bot - Domain Overview . . . . .	43
4	Spot The Bot - Win Rates . . . . .	45
5	Spot The Bot - Survival Features . . . . .	47
6	Spot The Bot - IAA . . . . .	48
7	Spot The Bot - Annotation Efficiency . . . . .	50
8	Spot The Bot - Annotator Overview . . . . .	51
9	Spot The Bot - Examples . . . . .	53
10	Spot The Bot - Automated Ranking . . . . .	55
11	AutoJudge - Correlations . . . . .	63
12	AutoJudge - Human vs. AutoJudge scores . . . . .	65
13	AutoJudge - Answer Selection . . . . .	65
14	AutoJudge - Examples . . . . .	67
15	OTTA - Data Overview . . . . .	82
16	OTTA - Query Comparison . . . . .	83
17	OTTA - Complexity Comparison . . . . .	84
18	OTTA - Question Examples . . . . .	86
19	OTTA - Baseline Scores . . . . .	90
20	OTTA - Production Rules . . . . .	93
21	OTTA - Selection Rules . . . . .	94
22	OTTA - Evaluation Example . . . . .	96
23	OTTA - Data Overview . . . . .	97
24	OTTA - Pretrain Results . . . . .	101
25	OTTA - Rerank Results . . . . .	102
26	OTTA - Complexity Comparison . . . . .	103

# List of Acronyms

ACL	Association for Computational Linguistics
AMT	Amazon Mechanical Turk
AST	Abstract Syntax Tree
BNC	British National Corpus
CALL	Computer Assisted Language Learning
CFG	Context-Free Grammar
DG	Dependency Grammar
DTD	Document Type Definition
DS	Dialogue Systems
GAN	Generative Adversarial Network
HRED	Hierarchical Recurrent Encoder-Decoder
EACL	European Chapter of the Association for Computational Linguistics
LML	Lifelong Machine Learning
LSTM	Long short-term memory
MT	Machine Translation
NER	Named Entity Recognition
NLIDB	Natural Language Interfaces to Databases
NLP	Natural Language Processing
OCR	Optical Character Recognition
POS	Part-Of-Speech
QA	Question Answering
TEI	Text Encoding Initiative
UTF-8	Unicode Transformation Format (8-bit)
XML	eXtensible Markup Language

# 1 Introduction

This thesis treats the task of evaluating dialogue systems, a crucial step during dialogue system development, which is still unsolved. The underlying question that we ask when evaluating a dialogue system is if the dialogue was of sufficient quality. Different definitions of dialogue quality depend on the context and scope of the dialogue system under consideration. We define three types of dialogue systems in research: task-oriented systems, conversational dialogue systems (also known as chit-chat bots), and question-answering dialogue systems [Deriu et al., 2020b].

- Task-oriented dialogue systems are developed to solve a task. For instance, book a flight from Madrid to London on a given date. They are characterized by a highly structured dialogue, which does not allow for dialogues out of scope.
- Conversational dialogue systems (also called chit-chat bots or social bots) are developed to engage the user in a conversation. There is no strictly defined goal. They are characterized by conversations that allow for more variety in different topics, such as talking about the weather or trivia.
- Question-Answering (QA) dialogue systems are developed to answer questions by the user. There are different types of question-answering systems. For instance, extractive QA systems that allow for questions about any topic [Choi et al., 2018; Campos et al., 2020] (e.g. Google <sup>1</sup>). Natural Language Interfaces to Databases (NLIDB) that answer questions on the contents of a structured database [Affolter et al., 2019].

For completeness sake, we note that this thesis focuses on dialogue systems developed in a research context. In an industrial setting, the different dialogue system types are more integrated than in research settings. For instance, Siri <sup>2</sup> acts as a conversational dialogue system and a question-answering system. Another example is XiaoIce [Zhou et al., 2020] that allows for all three types of dialogue system behavior depending on the users' intent. However, these systems are out of scope for this thesis. Here

---

<sup>1</sup><https://www.google.ch/>

<sup>2</sup><https://www.apple.com/es/siri/>

we focus solely on dialogue systems that are developed in the research context. Furthermore, we limit our discussions to dialogue systems that work with written English texts. We exclude dialogue systems that work with different modalities (e.g., speech or gestures), and we exclude dialogue systems trained in languages other than English. We limit this thesis’s scope to dialogue systems that are developed for research purposes for the English language. Further restrictions are mentioned at a more appropriate time.

Depending on the types of dialogue systems, different approaches to evaluations are needed. This is because they all serve different purposes.

**Task-Oriented Dialogue System** For instance, a task-oriented dialogue system is deemed high quality if it reaches its goal (e.g., selling a ticket to the customer) efficiently (i.e., with as few interactions as possible) [Schatzmann et al., 2007]. The evaluation of task-oriented systems is automatable as the aforementioned goals can be measured. Usually, a user simulation is developed to automate the evaluation process. One challenge is to model the user simulation as realistically as possible [Deriu et al., 2020b].

**Question-Answering Dialogue System** Question-answering dialogue systems are of high quality if they can answer many questions correctly. For instance, question-answering systems, which retrieve a span of text from a collection of texts to answer a question, are evaluated by their ability to retrieve the correct span. This is easily measured using F1 scores [Choi et al., 2018; Reddy et al., 2018; Campos et al., 2020]. If the question-answering system is tasked to answer questions over a structured database, the success can be measured using the correctness of the result set or by analyzing the query produced by the system [Yu et al., 2018b; Deriu et al., 2020a].

**Conversational Dialogue System** On the other hand, conversational dialogue systems are hard to evaluate [Liu et al., 2016; Deriu et al., 2020b]. The fundamental problem lies in what defines a high-quality dialogue and how to operationalize the definition. That is, what is the metric and what is the method to measure this metric? For task-oriented systems, the metric is efficiency and measured by the number of turns needed to reach a goal. For conversational dialogue systems, there exist various metrics that stem from different viewpoints. However, there is not yet an evaluation method, which can measure the metrics reliably and efficiently.

The oldest viewpoint is to measure the intelligence of a dialogue system (or artificial intelligence), a view proposed by Turing measured by the Turing Test [Turing, 1950]. Here, a human judge is tasked to talk to a system, which is either a dialogue system

or a human that operates a chat interface. When a dialogue system can convince the judge that it is a human, the dialogue system passes the test.

A different viewpoint stems from the industry that aims to maximize user engagement. Thus, the dialogue system is evaluated regarding its ability to engage the user in a long conversation. Although the metric (i.e., engagingness) is clear, the method is not straightforward. One possibility is to measure the length of the conversations [Zhou et al., 2020] following the intuition that a long conversation means that the dialogue system is engaging. A different method is to ask humans to interact with the dialogue system and rate if they found the conversation to be engaging [Dinan et al., 2020]. However, it is not clear for each method if it measures the engagingness of the conversation or something different.

Another viewpoint stems from the linguistic perspective. Most notably, Grice [1975] introduced four maxims of conversation:

- Maxims of Quantity (Informativeness): Ones' contribution should be as informative as required, not more and not less.
- Maxims of Quality (Truthfulness): Ones' contribution should be true. That means one should not say what he believes to be false or that for which there is no evidence.
- Maxims of Relation (Relevancy): Ones' contribution should be relevant to the current context of the conversation.
- Maxims of Manner (Clearness): Ones' contribution should be easy to understand, unambiguous, brief, and orderly.

These maxims were introduced for general conversations and not specifically for dialogue systems. However, a good dialogue system should adhere to these maxims. Current conversational dialogue systems struggle with all these maxims. For instance, they are not truthful, as they tend to change personality from one utterance to the next [Zhang et al., 2018]<sup>3</sup>. Although these maxims serve as a good guideline for conducting a good conversation, they are not easy to measure. For instance, letting humans judge if a contribution is relevant to the context might yield different results depending on the judge. This leads the evaluations to suffer from low inter-annotator agreement making the evaluation unreliable.

**Example** To better showcase the problems of evaluation for conversational dialogue systems, we consider Table 1. This is a conversation with two interlocutors, P1 and

---

<sup>3</sup>Here, we note that we want to avoid the philosophical discussion of what it means to know the truth and if a dialogue system can know what is true or not.

Interlocutor	Utterance
P1	Why are you so obsessed with online shopping?
P2	I just do not like crowds of stores.
P1	You can choose to go to stores where people do not crowd a lot!
P2	Why are you so concerned with what I like?
P1	You do not have to restrict your life always on one side.
P2	Not shopping in stores is not restrictive. I can go ice skating all winter long!
P1	You have to learn to bear with life situations for the time being.
P2	I disagree. My parents practice law and I have dealt with enough.

Table 1: Example of a conversation between P1 and P2.

P2. Here, P1 asks P2 about their shopping preferences, and they talk about it. How do we rate this conversation? A critical piece of information is the context. If this dialogue is between two humans, then we rate it differently than when it is between a human and a chatbot since, at the time of writing, we expect a dialogue system to perform worse than a human. If we rate this dialogue with regard to its human-likeness, one might be tempted to state that P1 and P2 could be humans. However, it is not clear what criteria are used to assess this. On the other hand, if we ask about the level of engagingness of P1, then we might rate P1 as not very engaging since P2 seems to be more and more annoyed by P1s' behavior. If we apply Grices' maxims, we can perform a different analysis. We might rate P2s' level of *informativeness* as high but not perfect, as in the second to last exchange, P2 starts to talk about ice skating (maybe to divert from the uncomfortable conversation). The same could be said with respect to *relevancy*. We might rate P1s' informativeness as low as P1 delivers more information than is required by P2. However, we note that these are the author's judgments, and the reader might conclude differently.

**Inefficiency.** Another aspect is that the evaluation is inefficient. In most cases, human effort must be leveraged. For instance, humans must interact with the dialogue system, which is a costly and time-consuming task. It is hard to let humans interact with dialogue systems without any instructions and training [Dinan et al., 2020]. Thus, human evaluation needs to be carefully planned, and the judges must be trained to interact with the system. This increases the cost of the evaluation.

A final aspect that we discuss in this thesis is about the time of evaluation. Ideally, a dialogue system would leverage the interaction with humans to improve its capabilities. For instance, in a task-oriented dialogue system, the online feedback



can be leveraged to improve a system after deployment [Gašić et al., 2011]. Here, the user states if the task has been successfully solved, which is then translated to a reward function. However, this approach is not easily applicable for the other two types of dialogue systems. For instance, assume a question-answering system for structured data. If the user asks for all the customers who ordered a movie before a given date, the system returns a list of customers. However, the user cannot be sure if the system performed the task correctly since the system might have inserted a wrong date or the wrong movie title.

This Thesis is concerned with two main questions: first, how can the cost of evaluation be reduced, and second, how can a dialogue system leverage live user feedback to self-assess its performance? We restrict the scope to handle neural conversational dialogue systems developed in the research context and question-answering systems that work over structured data, such as SQL databases. The latter systems are also referred to as Natural Language Interfaces to Databases (NLIDB).

## **1.1 Problem Statement for this Thesis**

### **1.1.1 Increased Efficiency of Evaluation**

The main issue in evaluating conversational dialogue systems lies in their unreliability and high cost and time intensity. The long-term goal is to develop an automated method that can reliably evaluate a dialogue system. For task-oriented dialogue systems, there already exist methods for evaluation, which reduce human involvement. Question-answering dialogue systems are also evaluated based on automated methods, which measure the response’s correctness to a given question. However, for conversational dialogue systems developed for engaging conversations with humans, there are no reliable methods for evaluation. In fact, automated methods (usually based on BLEU score) are shown not to correlate with human judgments at all [Liu et al., 2016]. Even human-based evaluations tend to be unreliable because the aforementioned metrics (e.g., Grices’ maxims) are hard to translate into a reliable evaluation method. This leads human-based evaluations to be prone to low agreement scores and thus, resulting in unreliable comparisons. Thus, the first problem statement in this Thesis is: how can the costs of evaluating a conversational dialogue system be reduced while ensuring that the evaluation is robust and reproducible?

### 1.1.2 Unreferenced Evaluation after Deployment

The second problem statement is concerned with the evaluation after deployment, where there is no access to labeled data. In machine learning applications, evaluation is often tailored towards the development phase of the algorithm. However, during the deployment phase, the measurement of the live performance is of great interest. Assessing the deployed model’s performance would lead to the ability of live monitoring and opens up to the possibility of automated improvement over time. The main issue is the lack of a gold standard, which can be used for comparison. Thus, the need for so-called unreferenced metrics arises, that is, metrics that do not need a gold reference. Since dialogue systems are designed to interact with humans, the interactive nature of the conversation can be leveraged to automatically get feedback, which can be used to improve the system. Especially for Natural Language Interfaces for Databases (NLIDB), unreferenced methods are of great value since they remove the need to employ costly SQL experts for data annotation. Thus, the second problem statement in this Thesis is: how can NLIDB systems be evaluated without the need for a gold standard, and can this method be used to improve the system automatically?

## 1.2 Thesis Contributions

The contributions of this Thesis focus on conversational dialogue systems and NLIDBs. This Thesis adds the following contributions:

**Bot-Bot Talk as basis for evaluation.** We present a novel paradigm for evaluating conversational dialogue systems based on automatically sampled dialogues between two dialogue systems (this also includes self-talk, i.e. when a bot talks to itself). We present two applications, which are based on bot-bot talks. First, *Spot The Bot* is a framework for human evaluation, where humans are shown dialogues between dialogue systems and need to decide for each interlocutor if it is a human or a bot. The second application is *AutoJudge*, a trained metric, i.e., a metric trained on examples of rated dialogue turns. These judgments are then used to train a regression model, which learns to judge dialogue turns automatically. The bot-bot talk paradigm allows for evaluating the multi-turn behavior of dialogue systems without generating dialogues between humans and dialogue systems. Thus, both proposals improve efficiency.

**Comparative Evaluation of Dialogue Systems.** With *Spot The Bot*, we introduce a novel human evaluation protocol for evaluating conversational dialogue sys-

tems efficiently and reliably. The evaluation is based on conversations between bots allowing for a comparative evaluation, which is shown to be more robust. Given a pool of bots, we randomly sample a set of conversations for each pair of bots through bot-bot talk and let humans find out for each interlocutor if it is a bot or a human. We can then create a ranking for the bots in the pool, which we show to be robust. Since current conversational dialogue systems are not yet human-like enough, we add a time component, which allows us to compare which dialogue system can maintain a human-like appearance for the most amounts of turns.

**Novel Data Collection Methodology for NLIDB.** We introduce a novel data gathering procedure for NLIDB, which increases the efficiency and the amount of content covered in a database. Traditionally, the annotation process is performed manually by letting SQL experts produce pairs of questions and SQL queries or automatically generate question and query pairs based on manually generated templates. We propose to invert the annotation procedure by sampling the queries from a context-free grammar and letting humans write the corresponding question. Our approach produces highly complex questions, which most semi-automated approaches for data generation could not do. To this end, we introduce a novel representation for the SQL queries based on the logical execution plan found in relational databases. It reflects the idea to represent a query as a sequence of operations, where the result of one operation is used as input for the next operation.

**Evaluation after Deployment for NLIDB systems.** We propose an unreferenced evaluation method (i.e., an evaluation method that does not rely on manually annotated data) for NLIDB based on back-translation and textual semantic similarity. To this end, we introduce a rule-based back-translation procedure to represent a query in natural language. We apply this to back-translate the generated query of our NLIDB system. Given the original question and the back-translated question, we apply a textual semantic similarity. This procedure is used as a proxy for evaluating if the NLIDB system correctly answers a question. We show that it can be used as a proxy to evaluate the system.

**Improvement of the NLIDB system.** We propose two applications of the back-translation and semantic similarity evaluation to improve the system’s output based on these results. The first application automatically enhances the training set with automatically generated pairs of questions and queries and uses this synthetic dataset for pretraining. The second application is to rerank the hypothesis of the beam search by semantic similarity scores. We show that the combination of these two applications yields significant improvements.

## 1.3 Outline

First, we review the background of research on the evaluation of dialogue systems in chapter 2. Since the evaluation is tightly coupled with the dialogue systems themselves, we also introduce research on dialogue systems and the relevant corpora for the thesis. Chapter 3 introduces the novel paradigm, namely, bot-talk, as the basis for evaluation. We show its application on both human evaluation and automated evaluation. Chapter 4 introduces the more efficient data collection procedure, the unreferenced evaluation protocol, and the improvement strategy. In chapter 5, we conclude the thesis and present an outlook on future research.

## 2 Dialogue Systems

In this section, we introduce the relevant background for dialogue systems and their evaluation. Since we introduce only the relevant parts for the thesis, we point the reader to [Gao et al., 2018; Deriu et al., 2020b] for a more exhaustive overview of the field (especially for task-oriented systems and question-answering systems). Note that this section extends [Deriu et al., 2020b] with related work on Natural Language Interfaces to Databases (NLIDB).

We first give a general overview of dialogue systems, then we present the background on conversational dialogue systems and question answering dialogue systems, or more precisely, Natural Language Interfaces to Databases (NLIDB).

### 2.1 General Overview

We begin by introducing the different types of dialogue systems encountered in current research, and then we introduce the general structure of the dialogues, which we use throughout the rest of the thesis.

#### 2.1.1 Classification

Dialogue Systems (DS) are computer systems that are capable of interacting with humans via conversation. The research on Dialogue Systems can be broadly categorized into three different types: task-oriented dialogue systems, conversational dialogue systems, and question-answering dialogue systems<sup>1</sup> [Gao et al., 2018; Deriu et al., 2020b]. The dialogue system types can be characterized concerning various features.

Table 2 shows an overview of the types and their features. The dialogue systems

---

<sup>1</sup>Some literature classifies dialogue systems only as task-oriented DS and conversational DS [Serban et al., 2018; Chen et al., 2017; Jurafsky and Martin, 2017]. However, the classification into three types makes more sense for our purposes since question-answering DS cannot be placed in either of the two categories.

	Task-oriented DS	Conversational Agents	Interactive QA
Task	Yes - clearly defined	No	Yes - answer questions
Dial. Structure	Highly structured	Not structured	No
Domain	Restricted	Mostly open domain	Mixed
Turns	Multi	Multi	Single/Multi
Length	Short	Long	-
Initiative	Mixed/ system init	mixed/user init	user init
Interface	multi-modal	multi-modal	mostly text

Table 2: Characterizations of the different dialogue system types. Taken from Deriu et al. [2020b].

are classified according to the following features: if they have a clearly defined task to solve, if the dialogue is structured or open, if the domain is open or restricted, if the dialogue takes a single turn only or spans over multiple turns, if the dialogue should be long or short, who takes the initiative, and how the dialogue takes place.

Task-oriented dialogue systems are characterized by a strictly structured conversation dictated by predefined dialogue acts and slot-value pairs. A task-oriented DS’s goal is to efficiently solve a task for the user (e.g., book a reservation in a hotel for a given date). A more open conversation style characterizes conversational dialogue systems; there is no goal to be reached other than engaging the user. Moreover, the conversations do not follow a structured flow, as is the case for task-oriented systems. Multi-turn dialogues are characterized by keeping track of the context, whereas single-turn dialogues do not remember previous exchanges. While the aforementioned dialogue system types work on multiple turns, question-answering systems are often built only for a single exchange dialogue. Although, recent research proposed datasets for multi-turn question-answering systems [Reddy et al., 2018; Choi et al., 2018; Yu et al., 2019; Campos et al., 2020]. Question-answering systems do have a single task to solve, namely, to answer the users’ questions. This lies in contrast to task-oriented systems, where the goals are more diverse and complex. This thesis focuses on dialogue systems that work on written dialogues since spoken dialogue systems or multimodal dialogue systems are out of scope.

The different characteristics of the dialogue system types are tightly coupled with the evaluation procedures used for various dialogue systems. Since each dialogue system serves a different purpose, the evaluation should be adapted to this purpose [Paek, 2001]. For instance, task-oriented systems are often evaluated on their ability

to correctly and efficiently reach the user goal. This procedure cannot be applied to conversational dialogue systems, which are evaluated regarding how engaging they are or how appropriate their responses are to a given conversational context. Question-answering systems can be evaluated with regard to the correctness of their answers. This thesis focuses only on conversational dialogue systems and question-answering systems (more precisely, NLIDB systems, which we introduce later).

### 2.1.2 General Dialogue Structure

Here, we briefly introduce the terminology for the dialogue structure, which is used throughout the thesis’s remainder. We extend the definition by Deriu et al. [2020c]. In this thesis, we assume that in each dialogue, there are two interlocutors. A dialogue  $D$  is structured as a finite sequence of exchanges  $D = e_1, e_2, \dots, e_n$ , where each exchange is composed of two turns  $e_i = t_1^i, t_2^i$ . Each turn is an utterance of one of the two interlocutors composed of a sequence of tokens  $t_j^i = w_1, w_2, \dots, w_{m_{i,j}}$ . Thus, we assume that the conversations are a sequence of turns, where the two interlocutors alternate. For the scope of this thesis, we assume that the conversation is text-based and not multi-modal (i.e., audio, video, gestures). We also assume that there are no barge-ins. That is, each interlocutor finishes its utterance without interruption.

## 2.2 Conversational Dialogue Systems

Conversational dialogue systems are not built to solve a specific task; instead, they are developed for open conversations with humans. There are different subcategories of conversational dialogue systems. There are open domain chit-chat bots developed for open-ended conversations about any topic [Zhou et al., 2020]. Another branch of research is concerned with so-called "Virtual Humans", which are simulations of human behavior designed for more specific purposes. For instance, a virtual Holocaust survivor can preserve the memories of historical events, which can be interactively accessed [Traum et al., 2015]. The two subcategories of conversational agents also differ in their implementations and research approaches. On the one hand, the open domain chit-chat bots are usually built as either a large-scale neural network trained in an end-to-end fashion [Roller et al., 2020] or as an ensemble of response generation systems, which are coordinated by a selection mechanism [Serban et al., 2017b; Zhou et al., 2020]. On the other hand, Virtual Humans are developed by selecting the most appropriate answer in a pool of pre-recorded answers. In most cases, Virtual Humans work on a much smaller scale.

This thesis focuses on open-domain neural conversational dialogue systems, usually trained on large amounts of dialogue data. The models, which we introduce in this section (except for the rule-based models), are used in chapter 3 as models under evaluation. We deem it useful for the reader that we provide a basic overview of the systems, which we use as objects of evaluation. However, we refer the reader to the respective papers for a more detailed explanation of the various models.

### 2.2.1 Neural Conversational Dialogue Systems

The historical evolution of general-purpose open-domain conversational dialogue systems can be split into three phases:

- **Rule-Based Systems.** In the early stages, the conversational dialogue systems were based on rules that determine the system’s behavior. A prominent exponent of this is ELIZA [Weizenbaum, 1966], which works on simple pattern recognition and rules that apply transformations to the user’s input. Another example is PARRY [Colby, 1981], a chatbot that models a paranoid mind. Today, rule-based systems are mostly used in the context of industrial chatbots. For instance, DialogFlow<sup>2</sup> allows the dialogue engineer to define rules on how the chatbot should handle certain requests. The rule-based systems are enhanced with natural language understanding (NLU) units, which extract the users’ intent.
- **First Neural Phase.** In this phase, the conversational dialogue systems were modeled using the sequence-to-sequence architecture [Sutskever et al., 2014]. These systems are trained on large amounts of conversational data.
- **Second Neural Phase.** Recently, the large-scale language models based on large-scale neural networks and pre-trained on large amounts of textual data (e.g., WebText) were introduced [Devlin et al., 2019; Radford et al., 2019]. The latest neural conversational dialogue systems fine-tune these language models on conversational data [Zhang et al., 2020; Roller et al., 2020].

The transition from rule-based systems to neural network-based systems was possible due to the large amounts of textual data, which became accessible through the increased usage of the internet and increased computational power, which enabled the training of neural networks on such large datasets.

The different types of dialogue systems differ in their capability of generalizing to

---

<sup>2</sup><https://dialogflow.cloud.google.com/cx/projects>



unseen situations and their capability to be specific to the context of the conversation. For instance, ELIZA could handle most user inputs. However, it worked with unspecific answers. Furthermore, rule-based systems focus only on a single domain or a very narrow scope. On the one hand, neural dialogue systems are able to handle broader scope, and they handle a larger variety of topics. On the other hand, they suffer from repetitive and generic answers and are often unable to remain consistent throughout the conversation. The dialogue systems built by fine-tuning large-scale language models are shown to return more specific responses to the user's input and, in some cases, uphold consistency and adhere to the context for more exchanges.

A different distinction of neural conversational agents can be made on the level of dialogue strategy:

- **Generative Methods.** These dialogue systems usually employ a decoder, which given the conversational history, generates the response by repeatedly sampling tokens from a vocabulary. These systems usually follow the encoder-decoder paradigm [Sutskever et al., 2014].
- **Ranking Based Methods.** These dialogue systems are modeled to retrieve human written utterances from a pool of possible answers. Usually, there is a database of utterances from which the dialogue system can choose. The goal is to rank the candidate utterances by relevance and return the most appropriate utterance to the user [Lee et al., 2009; Lowe et al., 2017b].

In the following, we introduce the generative and ranking-based methods, with a focus on neural approaches.

## **2.2.2 Generative Approach**

We distinguish between two types of generative approaches: the encoder-decoder-based approach, which we attribute to the first neural phase, and the language model-based approaches, which we attribute to the second neural phase.

### **2.2.2.1 Encoder-Decoder Based Approaches**

The generative approaches are characterized by following the encoder-decoder architecture inspired by the machine translation literature. The idea is to model dialogues as a translation task [Ritter et al., 2011]. The encoders and decoders are usually based on Recurrent Neural Networks (RNN) or Long short-term memory networks (LSTM) [Hochreiter and Schmidhuber, 1997]. The output of the encoder

is the last hidden state after the sequence of words has been processed. This hidden state is then used to initialize the hidden state of the decoder. Thus, the decoder is conditioned on both the encoder's hidden state and the previously generated tokens at each step. The first applications of the Sequence to Sequence (Seq2Seq) architecture to dialogue modeling were introduced by Vinyals and Le [2015] and Shang et al. [2015]. The novelty was that these models were trained using large amounts of data ( $\approx 10^6$  conversations).

The Seq2Seq model can be described in terms of the conditional probability of turn  $t_2^i = w_{1:m_{i,2}}^{i,2}$  given the previous turn  $t_1^i = w_{1:m_{i,1}}^{i,1}$ . where  $m_{i,j}$  is the number of tokens of turn  $t_j^i$  and  $w_k^{i,j}$  denotes its  $k$ -th token.

$$p(t_2^i | t_1^i) = \prod_{k=1}^{m_{i,2}} p(w_k^{i,2} | w_{<k}^{i,2}, w_{1:m_{i,1}}^{i,1}) \quad (2.1)$$

Thus, the prediction of the next token  $w_k^{i,2}$  depends on  $w_{<k}^{i,2}$  (i.e. the  $k - 1$  previous tokens of turn  $t_2^i$ ) and all the tokens in turn  $t_1^i$ .

The subsequent research tackled two main issues. First, taking the context into account, that is, referring to previous utterances, avoiding sudden jumps in topics, and avoiding being inconsistent. For instance, most models do not have a coherent persona, which results in inconsistent answers to the same questions (e.g., E1: "What is your job?" E2: "I am a student.", ..., E2: "I am a mechanic."). The second problem is that the models tend to produce generic answers (e.g., "I don't know"). This issue is a result of producing the safest utterance to any given context.

**Context.** The context of the conversation is usually defined as the conversation history (i.e., the previous turns). There are different approaches to include the conversation history into the neural architecture. One approach was proposed by Sordoni et al. [2015], which is to model the context as a bag-of-words, which is used to condition the generation process. Thus, the training data for the neural network is a triple of context, message, and response. A related approach is to concatenate the last  $n$  turns to a long sequence, which is then fed as input to the encoder [Shang et al., 2015].

In terms of the conditional probability, the generation of turn  $t_2^i$  is now also dependent on all the previous exchanges  $e_{1:i-1} = t_1^1, t_2^1, \dots, t_1^{i-1}, t_2^{i-1}$  as well as the previous turn  $t_1^i$ . Let  $w^{<i-1}$  denote all tokens that appear in  $e_{1:i-1}$ , then the conditional probability is expressed as follows:

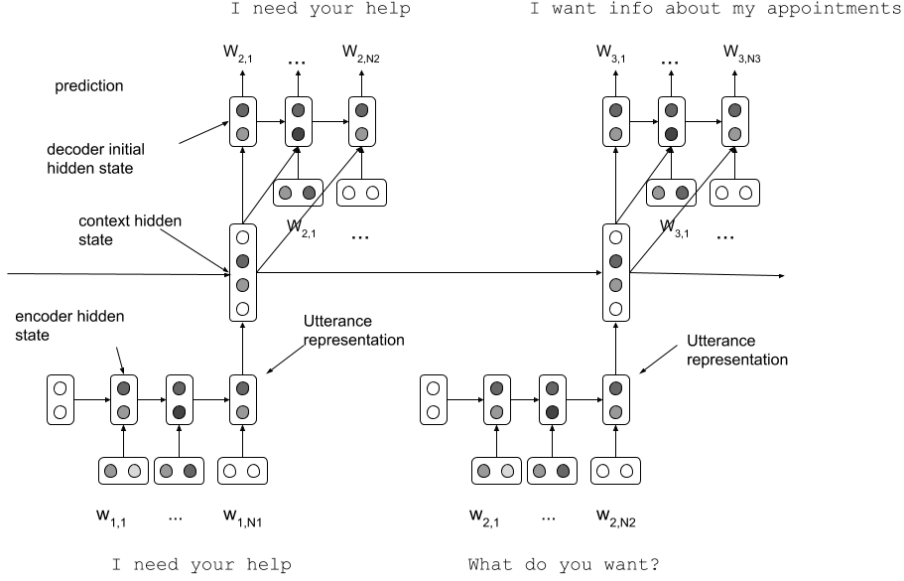


Figure 1: The HRED architecture by Serban et al. [2016]. It consists of two encoders: a token-level encoder, which encodes each turn, and a context-level encoder, which creates the latent representation of the conversation context. The decoder is conditioned on the latent representation of the conversation context and generates the response utterance. Figure taken from Deriu et al. [2020b]

$$p(t_2^i | t_1^i, e_{1:i-1}) = \prod_{k=1}^{m_{i,2}} p(w_k^{i,2} | w_{<k}^{i,2}, w_{1:m_{i,1}}^{i,1}, w^{<i-1}) \quad (2.2)$$

A different approach is to model the conversation as a hierarchy of sequences: the dialogue is a sequence of turns, and each turn is a sequence of tokens [Serban et al., 2016]. The hierarchical recurrent encoder-decoder (HRED) model comprises two encoders: the turn encoder and the context encoder (see Figure 1). The turn encoder is a recurrent neural network, which encodes each turn of the context separately. Thus, it computes a hidden state of each turn. This sequence of hidden states is then fed into the context encoder, which produces the hidden state, which is then used to condition the decoder.

In contrast to the previous models, HRED models the sequence of turns in a dialogue and not just the next turn. Thus, the probability of the dialogue  $D$  is expressed as follows:

$$p(D) = p(e_{1:n}) = \prod_{i=1}^n p(t_2^i | t_1^i, e_{1:i-1}) = \prod_{i=1}^n \prod_{k=1}^{m_{i,2}} p(w_k^{i,2} | w_{<k}^{i,2}, t_1^i, e_{1:i-1}) \quad (2.3)$$

In contrast to equation 2.2, it models the conversation in a hierarchical structure by conditioning the generation of the next token on the previously generated tokens and the latent representations of the previous turns.

The HRED architecture serves as the basis for more complex architectures. For instance, the multi-resolution recurrent neural network (MrRNN) [Serban et al., 2017a] adds multiple encoders to model the conversation at different levels of granularity. More specifically, the second encoder reads the conversation at the level of entities that appear in the conversation.

**Generic Responses.** There are different approaches to remedy the problem of generic responses. There are two main observations: first, since the perplexity is optimized, general responses are favored since they yield low perplexity scores [Li et al., 2016a]. The second observation is that variability stems only from the generation process, which is conditioned only on the context representation and the previously generated tokens [Serban et al., 2017c]. Due to the vanishing gradient effect, the conditioning on the previous tokens is valued more strongly. Based on these observations, two approaches to remedy this problem arise: adapting the loss function and changing what the decoder is conditioned upon.

Li et al. [2016a] propose two loss functions based on maximum mutual information to increase diversity by penalizing generic responses and promoting more diverse responses. The first loss is based on a language model, which is used to penalize utterances with low perplexity:  $p(t_2^i | t_1^i, e_{1:i-1}) - \lambda p(t_2^i)$ , where  $\lambda < 1$  regulates the influence of the so-called anti-language model. Too high  $\lambda$  values would lead to ungrammatical utterances, as these utterances are assigned higher perplexity scores by the language model. The second loss function is based on back-translation:  $(1 - \lambda)p(t_2^i | t_1^i, e_{1:i-1}) + \lambda p(t_1^i, e_{1:i-1} | t_2^i)$ . Thus, the second term denotes the probability of the context given the generated response.

Serban et al. [2017c] introduce VHRED, an extension of the HRED architecture that adds a stochastic random variable in the generation process, which is trained to predict the next context vector. The generation process is adapted to work in two phases: first, the latent variable is sampled, and then the output is generated. The latent variable encodes the long-term goal, i.e., how the next context vector should look like, making the generation process more dependent on the context, not only on the previously generated tokens. Thus, the gradient does not vanish, and the context’s signal is valued more strongly during the backpropagation step.

### 2.2.2.2 Language Model Approaches

Large-scale language models ( $\approx 10^9$  parameters), which are trained on large amounts of data ( $\approx 10^8$  documents) were recently introduced. The most prominent are BERT by Devlin et al. [2019] and GPT-2 by Radford et al. [2019]. These language models achieved great empirical results on many different natural language processing tasks. Recently, these approaches have been applied to dialogue systems as well. These approaches are similar to the previously mentioned approaches, i.e., they follow the encoder-decoder structure. However, they differ in the number of parameters used and the pre-training.

DialoGPT by Zhang et al. [2020] is a GPT-2 model fine-tuned on 150M dialogue instances from the Reddit corpus <sup>3</sup>.

The context is modelled by simply concatenating the whole dialogue history, thus, the generation process is modelled as  $p(w_k^{i,2} | w_{<k}^{i,2}, w_{1:m_{i,1}}^{i,1}, w^{<i-1})$ . That is, the next token depends on the previously generated tokens and the tokens in the context. Due to the nature of transformer models, which work with multiple self-attention layers, long-term dependencies can be modeled efficiently. The empirical results show that the responses generated by DialoGPT are diverse and relevant to the context. The diversity was measured as the ratio of distinct uni- and bi-grams and the total number of generated words. The relevancy to the context was measured by a human evaluation. The humans rated the relevance of a generated response on a 3-point Likert scale.

### 2.2.3 Retrieval Approach

Retrieval or ranking-based models work by ranking a large set of candidate responses by relevance. Neural approaches often work by encoding both the context and the candidate utterance, thus, creating latent representations, which are then compared using a similarity score. The differences between the models are dictated by the encoders used and by the loss function's modeling (i.e., the similarity function).

The Dual Encoder by Lowe et al. [2017b] uses the HRED encoders to encode the context and a LSTM-encoder to encode the candidate utterance. The similarity between the context embedding  $c \in \mathbb{R}^d$  and the candidate embedding  $r \in \mathbb{R}^d$  is computed as  $p(\text{rel} = 1 | c, r, M) = \sigma(c^T M r + b)$ , where  $M \in \mathbb{R}^{d \times d}$  is a trainable parameter. An alternative similarity measure is based on the concatenation of the context and response vectors, with a subsequent logistic regression layer:  $p(\text{rel} =$

---

<sup>3</sup><https://files.pushshift.io/reddit/>

$1|c, r, M) = \sigma(M[c; r] + b)$ , where  $M \in \mathbb{R}^{2d \times 2d}$  is a trainable parameter [Deriu and Cieliebak, 2017].

Similar to the generative models, more recent approaches employ large-scale language models to encode the context and candidate response. There are two main approaches to applying fine-tuned language models: Bi-Encoders and Cross-Encoders. The Bi-Encoder is most similar to the Dual Encoder, where the context and the candidate are encoded using fine-tuned language models [Dinan et al., 2019]. The Cross-Encoder jointly encodes the context and candidate utterance by concatenating the context and the candidate tokens, which creates a joint hidden representation that is then used to compute the score [Wolf et al., 2019]. Bi-Encoders tend to be faster during inference time since all the candidates can be pre-computed.

### 2.2.4 Mixed Approach

Blender is a large-scale transformer-based model proposed by Roller et al. [2020], which mixes the retrieval and generative approach. For this, so-called  $\alpha$ -blending is used, where during the training of the generative model, the gold utterance is replaced by a retrieved utterance in  $\alpha\%$  of cases. More precisely, in the first phase, a retrieval-based model is trained on large amounts of dialogue systems. The generative dialogue system is trained on pairs of contexts and gold responses from the training set in the second phase. However, in  $\alpha$  percent of cases, the gold response is replaced by a response, which the retrieval-based model provided for the given context. Both the retrieval and the generative model are based on large-scale language models, pre-trained on large amounts of dialogue data. The models are fine-tuned using a set of different corpora. Thus, Blender exhibits a variety of skills, such as being knowledgeable and empathetic.

## 2.3 Evaluation of Conversational Dialogue Systems

Evaluating a conversational dialogue system is still an unsolved problem. The main difficulty lies in the fact that it is hard to measure if a conversation adheres to the definitions of quality. For instance, it is not trivial to measure if an utterance is truthful or contains the right amount of information. This task is even challenging for humans to complete with a high agreement. This leads to unreliable evaluations.

The existing evaluation methods can be broadly separated into human-based evaluations and automated evaluations. The automated evaluation methods can be further

separated into methods for generative models and retrieval-based models. There are some general considerations to be made when evaluating a dialogue system [Gandhe and Traum, 2016].

- What is rated? There are different ways of how to present the output of a dialogue system to be rated. The main distinction here is between static and dynamic context. The static context is usually composed of human-human dialogues from the test set. Dynamic context, on the other hand, entails that the context is composed of human-bot conversations. This requires that humans interact with the bot to generate the context, which is time and cost-intensive. However, the dynamic context scenario is more realistic since the bot’s subsequent utterances are dependent on previous utterances. Furthermore, the behavior of the bot over multiple turns can be measured [Dinan et al., 2020].
- Who rates it? For both the static context and the dynamic context scenarios, the evaluation can be performed by a human or a machine. However, two types of humans can rate the dialogue: the user or a bystander. In the first case, the user who speaks to the dialogue system rates the system. This usually puts a higher cognitive demand on humans, making the evaluation less reliable [Evanini et al., 2008; Schmitt and Ultes, 2015]. On the other hand, letting bystanders rate the conversation reduces the mental strain on both the user who has to converse with the system and the judge.
- How to rate? The dialogues can be rated in various ways and according to different criteria specific to the application context. For instance, the dialogues can be rated on the turn-level or the dialogue level. Turn-level feedback is harder to obtain. However, it is useful for dialogue breakdown detection [Schmitt and Ultes, 2015]. Another distinction can be made regarding the granularity: is the whole behavior assessed, or is the evaluation only concerned with specific aspects of the dialogue systems’ behavior?

### 2.3.1 Human Based Evaluation

This section is based on and partially taken from Deriu et al. [2020c].

There are different approaches to human evaluation. The Turing Test is the first human evaluation of dialogue systems, which was proposed to measure the intelligence of a bot [Turing, 1950].

**Static context** evaluations are performed by presenting a human judge with a context sampled from a test set with the task to rate the utterance produced by the

system under evaluation. Thus, the contexts are human-to-human conversations. There are two methods of rating the single utterance: either on a Likert scale [Lowe et al., 2017a] or a comparative evaluation where the outputs of multiple different systems are ranked [Li et al., 2016b; Serban et al., 2017c]. The main issue with static contexts is that the dialogue system’s behavior over several turns is not considered Dinan et al. [2020]. Furthermore, a low inter-annotator agreement is often reported [Ghandeharioun et al., 2019].

**Dynamic context** evaluation is based on human-to-machine conversations allowing observing the dialogue systems behavior over several turns. In order to create these dialogues, human users are recruited to converse with the system. The dialogue is then rated on a Likert scale after the conversation [Ghandeharioun et al., 2019] either by the users or an objective bystander. The latter option reduces the mental strain on the user [Schmitt and Ultes, 2015]. However, this approach is not only time- and cost-intensive. It tends to produce useless dialogues if not done with care. For instance, in the ConvAI2 challenge, half of the collected conversations were of low quality [Dinan et al., 2020].

The evaluations based on Likert scales often suffer from high variance [Ghandeharioun et al., 2019], require normalization a posteriori, are prone to order effects and are less reliable than ranking-based ratings [Amidei et al., 2019a].

### 2.3.2 Automated Evaluation Methods

Since human evaluations are cost and time-intensive, which negatively impacts the research’s progress of conversational dialogue systems, much effort has been put into finding automated methods for their evaluation. The main distinction of the approaches is to either measure the human likeness of the dialogue system or the appropriateness of its responses [Deriu et al., 2020b]. Furthermore, retrieval-based models are evaluated differently than generative models, as methods devised for ranking evaluation can be applied. The evaluation of ranking-based systems is usually done using metrics from information retrieval. Recall @k ( $R@k$ ) is mostly used, which measures the fraction of relevant utterances in the top-k selected utterances [Lowe et al., 2016]. Automated metrics are evaluated with respect to their correlation to human judgments.



### 2.3.2.1 Human Likeness

The earliest method proposed for evaluating dialogue systems is the so-called Turing Test [Turing, 1950]. This test measures the dialogue system’s capability to mimic human behavior so well that it is indistinguishable from a human.

The closest framework for measuring the dialogue system’s capability to convince humans of its human likeness is adversarial learning [Goodfellow et al., 2014]. The generative adversarial networks (GAN) are composed of two modules: the generator and the discriminator. The generator learns to generate synthetic data, which is indistinguishable from human-generated data. The discriminator learns to detect synthetic data. The two components are jointly trained. Kannan and Vinyals [2016] trained a GAN on dialogue data and measured the dialogue systems quality based on the discriminator. If the discriminator gets high scores, it is unclear if it is due to a weak generator or the discriminator overfitting the generators’ output. Thus, the viability of the discriminator as a quality measure has to be determined. For this, Bruni and Fernandez [2017] compared the discriminators’ performance with human performance on the task of distinguishing human-generated utterances from synthetically generated utterances. Thus, they let humans annotate generated utterances to a given context as either human-written or synthetic and compare the human decisions to the discriminator’s decision. Their results show that humans not only have a low agreement (Fleiss  $\kappa = 0.3$  [Fleiss, 1971]), but they also achieve low scores on the task (Accuracy  $\approx 65\%$ ). The discriminator also achieves a low agreement with the human annotators and a similar accuracy score. However, it is unclear if the discriminator can automatically rate a dialogue system or automatically compare two dialogue systems. Furthermore, the evaluation is performed in a static context, which does not give the dialogue system enough turns to expose itself.

### 2.3.2.2 Appropriateness

Appropriateness is used to judge if the next utterance to a given context is appropriate. A strict definition of what constitutes an appropriate answer has never been provided. In each study, the term is defined in the context of the evaluation that is carried out. It often encompasses different finer-grained concepts such as coherence, fluency, relevance to the dialogue context, or factual correctness. However, it often depends on the goal of the evaluation.

Generally, there are two approaches, which have been used in recent literature:

word-overlap metrics and trained metrics.

**Word-overlap metrics** were originally devised by the machine translation and machine summarization communities and initially a popular choice to evaluate conversational dialogue systems for lack of alternatives. The most used metrics are the BLEU score [Papineni et al., 2002] and the ROUGE score [Lin, 2004]. The work of Liu et al. [2016] showed that neither of the word-overlap metrics correlates with human judgments.

**Trained metrics** are based on regression or classification models, which learn to rate a response utterance for a given context automatically. The metrics are trained on either labeled data, i.e., human ratings of dialogues or dialogue turns or trained on unlabelled data, such as language models or similarity measures. However, as discussed later, the low inter-annotator agreement scores cause trained metrics to be unstable.

Lowe et al. [2017a] propose an automatic dialogue evaluation method (ADEM), a neural network based on the dual encoder architecture optimized as a regression model to predict human ratings of utterances. The training data comprises 4-tuples: context, gold response, generated response, and the human rating (on a Likert scale from 1 to 5). ADEM predicts the human rating at test time given the context, the gold response, and the generated response. ADEM emulates the static context-setting where the goal is to rate the dialogue system on a Likert scale. The human ratings were collected using Amazon Mechanical Turk based on the static context evaluation method. Here, all the samples with low agreement scores are ignored. ADEM’s Pearson’s correlation to human judgments lies at 0.41 on the utterance level. Compared to the ROUGE score, whose correlation to human judgments lies only at 0.062. The correlation is also computed on the system level, where the system rating is computed as the average of all utterance scores by the same system. ADEM achieves a 0.954 Pearson correlation at the system level, which is high compared to the ROUGE score (0.268 correlation).

Tao et al. [2018] propose RUBER, a referenced metric and unreferenced metric blended evaluation routine, which mixes metrics that rely on a gold label and metrics that do not. The model is based on cosine similarity metrics, which work on the hidden representations of the context, the gold response, and the generated utterance. However, in contrast to ADEM, the method does not rely on manually labeled data to be trained on. The first submodule encodes the gold response and the generated response, on which the cosine similarity is computed to measure the relatedness of the gold response and the generated response. This encapsulates the observation that responses that are similar to the gold response are likely to be

good. The second submodule computes the cosine similarity between the hidden representation of the context and generated response’s hidden representation. This is based on the observation that responses related to the conversational contexts are likely to be good. RUBER achieves a Pearson’s correlation of 0.495, which is comparable to ADEM.

Mehri and Eskenazi [2020] propose URS, which is an unsupervised and reference-free evaluation method. It is based on fine-tuning transformer-based language models on dialogue data and uses these scores to evaluate various finer-grained aspects of a dialogue system. Two models are fine-tuned - first, a RoBERTa [Liu et al., 2019b] based language model, which is fine-tuned on dialogue training data. A generated response’s score is measured as the likelihood of the response under the fine-tuned RoBERTa model. This score is used to measure the naturalness and understandability of a generated response. The second model is based on a retrieval-based fine-tuning of RoBERTa, where a response is rated as either relevant or not for a given context. This score is used to measure the ability to maintain the context, being interesting, and using knowledge. The combined score is used to determine the overall quality of an utterance. It achieves a correlation of 0.481, which is comparable to the aforementioned methods.

There are two main issues with the current trained methods: first, it is shown that these methods are unstable, which leads them to be easily fooled [Sai et al., 2019], and second, these methods only work in a static context, which impedes them to measure the performance of a bot over several turns.

## 2.4 Natural Language Interfaces to Databases

Parts of this section are published in Deriu et al. [2020a].

Natural Language Interfaces to Databases (NLIDB) are question-answering dialogue systems, where the question is translated to a structured query (e.g., SQL or SPARQL), which is deployed against a database. There is a large amount of literature on NLIDB; here, we refer the reader to Affolter et al. [2019]. Here, we focus on the aspects, which are relevant to the thesis. There are different approaches to build an NLIDB system. NLIDB systems research comes from two directions: the database community and the natural language processing (NLP) community. The NLP community often refers to this task as Semantic Parsing. In recent years, the two directions converged.

The earliest systems are based on keyword searches over an inverted index [Simitsis

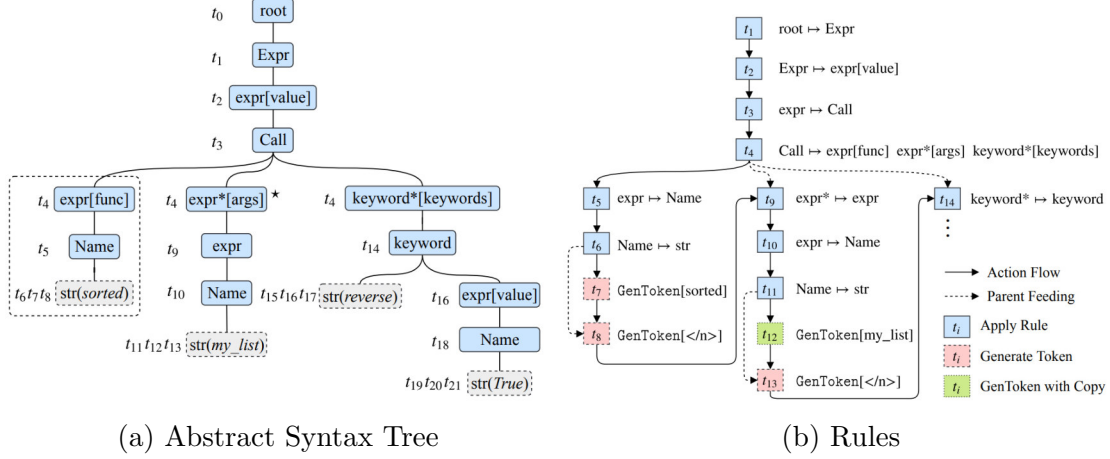


Figure 2: Taken from Yin and Neubig [2017]. The AST (a) and the set of rules (b) to generate the code: `sorted(my_list, reverse=True)` for the input "Sort my\_list in descending order". The sequence of actions in (b) is used to generate the AST. The output of the neural network is a sequence of actions.

et al., 2008; Blunschi et al., 2012; Bast and Haussmann, 2015]. While these systems work well for simple questions, they are limited in understanding more complex and ambiguous questions. Thus, these systems were enhanced with NLP technologies to handle these cases [Li and Jagadish, 2014; Saha et al., 2016]. The most recent approaches are based on neural networks, which follow the encoder-decoder architecture [Iyer et al., 2017; Basik et al., 2018; Cheng et al., 2019; Liu et al., 2019a; Guo et al., 2019].

SQL (and other programming languages) follow a context-free grammar, that is, a grammar where the rules follow the structure  $A \rightarrow \alpha$ , where  $A$  is a non-terminal symbol and  $\alpha$  is either empty or a string of terminal and non-terminal symbols. Most modern Semantic Parser approaches exploit this fact, which allows adapting the decoder accordingly. More precisely, the decoder is trained to generate an Abstract Syntax Tree (AST), which can then be deterministically converted to a program.

### 2.4.1 Grammar Net

Here we introduce *GrammarNet*, which is the NLIDB system, which we use for all our experiments in chapter 4. Here we provide the basic intuition needed for understanding the workings of *GrammarNet*. For a more technical description, we

refer the reader to Yin and Neubig [2017].

*GrammarNet* is a neural network, which incorporates the grammar in the decoder. They frame the code generation problem as translating a natural language description  $x$  to an AST  $y$ , which is then converted to the surface code  $c$ . The network comprises a natural language encoder, which creates a context-sensitive embedding  $h_i$  for each input token  $w_i$ . The encoder is based on a bidirectional LSTM [Hochreiter and Schmidhuber, 1997]. However, other neural network types are possible (e.g., Transformers). The second module is the decoder.

The decoder generates a sequence of derivation rules in the form  $A \rightarrow \alpha$ , which are then used to synthesize the AST. There are two types of rules: those where  $\alpha$  contains non-terminals, and those where  $\alpha$  is a terminal symbol. In the second case, the decoder needs to sample from the set of tokens. In the case of SQL, it would sample from the set of attributes, tables, or comparison operators. The rules are generated in a depth-first, left-to-right order fashion. Figure 2 (b), the generation process is visualized: at each step  $t_i$ , the information from the parent node and the last-sampled generation rule are used to condition the generation of the next action.

More formally, the decoder consists of an LSTM cell, which at each time-step uses the embedding of the previous action  $a_{t-1}$ , the context vector derived from the encoder using soft attention  $c_t$ , the embedding for the current node type  $n_t$ , and the state of the LSTM when generating the parent rule  $p_t$  to compute the next state  $s_t = LSTM([a_{t-1} : c_t : n_t : p_t], s_{t-1})$ . In order to sample the next action, two output embedding matrices are kept:  $W_R$  for the rules, and  $W_G$  for the terminal tokens, then the most probable rule or token is sampled based on the softmax distribution:  $a_t = \operatorname{argmax}(\operatorname{softmax}(W_R * \tanh(W * s_t + b)))$ . At inference time, beam search is applied to sample the sequence of rules. At each time step, a beam of size  $K$  is kept, which is then expanded and re-ranked based on the confidence score.

The Grammar Net by Yin and Neubig [2017] is used as the basis for most current approaches for translating natural language to programming languages [Guo et al., 2019; Wang et al., 2020] and is also used as the basis for the experiments in this thesis.

## 2.5 Evaluation of Natural Language Interfaces to Databases

The evaluation of NLIDB systems is more straightforward in that the results of the generated query and the gold query can be compared. Thus, an accuracy-based evaluation is possible. However, there are some intricacies in the evaluation methodology, which need to be addressed.

### 2.5.1 Metrics

Many systems do not predict the SQL query's values. That is, the values used to filter in the WHERE-clause are often not predicted [Yu et al., 2018b]. Thus, the automatically produced queries are not executable. The evaluation is performed by comparing the predicted SQL query's s and the gold query's surface form in these cases. There are two possibilities: either perform direct string comparison or perform a component-wise comparison [Yu et al., 2018b]. Direct string comparison is problematic as there are several syntactically different SQL statements, which are semantically equivalent. Thus, there is a high risk of creating false negatives.

**Component Matching.** Yu et al. [2018b] introduced component matching to evaluate the generated SQL query. Each query is decomposed into its components (SELECT, WHERE, GROUP BY, ORDER BY, KEYWORDS), and each component is represented as a bag of subcomponents, which are compared. For instance `SELECT COL1, MAX(COL2), COL3` is represented as the following set (COL1, MAX(COL2), COL3). Two components are then compared based on set comparisons, which are order-independent. Thus, there are fewer false negatives, and the evaluation is more fine-grained as it reveals where the error occurred.

### 2.5.2 Train-Test Split

Another issue in evaluation is how to create a train and test split, which measures the performance of the models' generalizability [Finegan-Dollak et al., 2018]. There are two main issues: how the train and test splits are generated and how the complexity of different queries is handled. The decision on how to generate the train and test split depends on what needs to be evaluated. There are two options on how to split the data: question-based split and query-based split.

**Question-based split.** In question-based splitting, the split is performed so that

each question-query pair appears only in the training data. In this scenario, many identical SQL queries may appear in both the training and test set. This splitting strategy measures the robustness of the system to answer known questions, which might be asked in a variety of ways (*"How many movies did Brat Pit produce?"* vs. *"Show me all the movies in which Brat Pit worked as the producer."*). Thus, if the training set contains all the questions, which users might ask about a database, then question-split will measure the model's robustness. However, it does not measure the model's capability to generalize to new types of queries. For instance, if the training set contains the questions *"In which movies did Keanu Reeves play in?"* and *"Which movies were released after 1992?"* then, the question split does not tell us how well the model performs on the composition: *"In which movies that were released after 1992 did Keanu Reeves play in?"*.

**Query-based split.** In order to evaluate the aforementioned scenario, the query split is needed. The query-split is generated by anonymizing the attributes, tables, and entities in the SQL query, such that SQL queries with the same structure are in the same equivalence class. Then, the split is performed such that the SQL queries of the same equivalence class are only present in either the training or in the test set.

Finegan-Dollak et al. [2018] showed that most datasets split the data according to the question-based approach. An evaluation of the state-of-the-art semantic parsers showed that most perform very well on the question-based split but show a very low performance on the query split.

### 2.5.3 Evaluation after Deployment

One major issue with machine learning systems, in general, is the evaluation after deployment. In this scenario, there are no gold labels to rely upon. Thus, the user and the service provider can never be sure whether the model makes errors. The challenge lies in finding ways to involve humans in the evaluation process through interaction or finding metrics that do not rely on a gold label.

There are two main questions about involving the human: how to design user feedback and how to exploit the feedback. Iyer et al. [2017] propose an interactive learning procedure, where the user is presented with the generated query's execution result. Since it is not obvious if a query is answered correctly, by showing the results only, visual aids were used to help the user decide if the question is answered correctly or not. They added two aids: highlighting the types of the entities identified in the question (e.g., *"In which movies did Keanu Reeves play in 2007?"* → *"In*

<b>User Utterance</b>	Before [September 10th], how many places was [Brad] employed?
<b>System NLG</b>	find employment [Brad] had, prior to [September 10th]
<b>User Feedback</b>	<i>Yes, but I also want you to count how many places Brad was employed at.</i>

Figure 3: Example of user feedback taken from Labutov et al. [2018]. The user asks a question, which the NLIDB system converts to an SQL query. The SQL query is translated to a natural language expression. The user sees that the system did not add the count operation and provides feedback accordingly. Note that the user utterance and the user feedback are grammatically incorrect, which might be because the data stems from a collection of emails.

*which movies did Keanu Reeves (PERSON) play in 2007 (YEAR)?*”). The second aid is to paraphrase the generated query. This means, if the generated SQL query also appears in the training set, then the associated utterance will be presented to the user. The users then decide for each question if the result is correct, wrong, or unsure. If the sample is labeled as correct, then the pair of question and generated SQL query is added to the training set. On the other hand, if the sample is deemed incorrect by the user, the question is sent to an SQL expert for annotation. The model is then retrained on the enhanced dataset every  $n^{th}$  question.

A different approach to designing and handling user feedback is to collect corrections from the users in natural language form. Labutov et al. [2018], and Elgohary et al. [2020] formulate this task as a triple of input question, generated SQL, and used feedback. After the user asked the question, the system generates the query and translates it to a natural language utterance using natural language generation. The user checks if the generated utterance matches his intent. If not, the user then provides feedback in natural language form. The system then uses the feedback to regenerate the query again. An example is depicted in figure 3.

## 2.6 Corpora

Here, we cover the most relevant existing corpora for this thesis. For conversational dialogue systems, we review the corpora’s content, whereas, for NLIDB, we also



Persona 1	Persona 2
I like to ski	I am an artist
My wife does not like me anymore	I have four children
I have went to Mexico 4 times this year	I recently got a cat
I hate Mexican food	I enjoy walking for exercise
I like to eat cheetos	I love watching Game of Thrones

[PERSON 1:] Hi  
 [PERSON 2:] Hello ! How are you today ?  
 [PERSON 1:] I am good thank you , how are you.  
 [PERSON 2:] Great, thanks ! My children and I were just about to watch Game of Thrones.  
 [PERSON 1:] Nice ! How old are your children?  
 [PERSON 2:] I have four that range in age from 10 to 21. You?  
 [PERSON 1:] I do not have children at the moment.  
 [PERSON 2:] That just means you get to keep all the popcorn for yourself.  
 [PERSON 1:] And Cheetos at the moment!  
 [PERSON 2:] Good choice. Do you watch Game of Thrones?  
 [PERSON 1:] No, I do not have much time for TV.  
 [PERSON 2:] I usually spend my time painting: but, I love the show.

Figure 4: Example dialogue in Persona-Chat. There are two predefined personas and the dialogue where the two personas get to know each other. Figure taken from Zhang et al. [2018].

focus on the methodology on how the corpora were created.

## 2.6.1 Conversational Dialogue Systems

For an extensive overview of corpora for dialogue systems, we refer the reader to [Serban et al., 2018; Deriu et al., 2020b].

### 2.6.1.1 Persona-Chat

The *Persona-Chat* corpus is composed of human dialogues, where two personas get to know each other [Zhang et al., 2018]. The data annotation is a three-phase process.

In phase one, crowdworkers are asked to create personas characterized by natural language descriptions composed of five sentences. In Figure 4, the two personas are described with five sentences, e.g., *"I like to ski"*. In this phase, 1155 personas were created.

Crowdworkers who write the dialogues tend to copy parts of the persona's description into the dialogue, which would make the task too easy for word overlap-based methods. Thus, in phase two, a second set of crowdworkers is asked to rephrase the persona's sentence. For this, they can make the description more general or more

A: I'm worried about something.  
B: What's that?  
A: Well, I have to drive to school for a meeting this morning, and I'm going to end up getting stuck in rush-hour traffic.  
B: That's annoying, but nothing to worry about. *Just breathe deeply when you feel yourself getting upset.*  
A: Ok, I'll try that.  
B: Is there anything else bothering you?  
A: Just one more thing. A school called me this morning to see if I could teach a few classes this weekend and I don't know what to do.  
B: Do you have any other plans this weekend?  
A: I'm supposed to work on a paper that'd due on Monday.  
B: *Try not to take on more than you can handle.*  
A: You're right. I probably should just work on my paper. Thanks!

Figure 5: Example dialogue in DailyDialog. Italic words are suggestions of B for how A could handle the situation. The underlined red words indicate emotions. Figure taken from Li et al. [2017].

specific. For instance, *"I am a big fan of Lionel Messi"* is paraphrased to *"I really like soccer"*.

In the last phase, crowdworkers are paired randomly and assigned a persona at random and asked to get to know each other by asking questions and answering the partner's questions. The dataset consists of almost 11k dialogues containing over 160k utterances.

This dataset is often used in evaluation campaigns [Dinan et al., 2020].

### 2.6.1.2 Dailydialog

The *Dailydialog* dataset is a human-written dialogue dataset covering many topics about human life. The dialogues are annotated with communication intents and emotion information [Li et al., 2017] (see Figure 5 for an example).

The dialogues are crawled from websites for teaching English. The dialogues are written by humans, and thus, more curated. The dialogues are more focused on a single topic since they span around 8 turns. These characteristics make it easier to train a dialogue system. The dataset contains 13k dialogues, with an average of 7.9 turns per dialogue. The dialogue turns are then manually labeled with communication intents (inform, question, directive, commissive) and emotions (anger, disgust,

<b>Label: Afraid</b> <b>Situation:</b> Speaker felt this when... "I've been hearing noises around the house at night" <b>Conversation:</b> <b>Speaker:</b> I've been hearing some strange noises around the house at night. <b>Listener:</b> oh no! That's scary! What do you think it is? <b>Speaker:</b> I don't know, that's what's making me anxious. <b>Listener:</b> I'm sorry to hear that. I wish I could help you figure it out	<b>Label: Proud</b> <b>Situation:</b> Speaker felt this when... "I finally got that promotion at work! I have tried so hard for so long to get it!" <b>Conversation:</b> <b>Speaker:</b> I finally got promoted today at work! <b>Listener:</b> Congrats! That's great! <b>Speaker:</b> Thank you! I've been trying to get it for a while now! <b>Listener:</b> That is quite an accomplishment and you should be proud!
---	---

Figure 6: Example dialogue in Empathetic Dialogues. The Speaker chooses an emotion from 32 options and writes about a situation in which they felt that emotion. Then the two workers talk about the situation. Figure taken from Rashkin et al. [2019].

fear, happiness, sadness, surprise).

### 2.6.1.3 Empathetic Dialogues

The *EmpatheticDialogues* dataset proposed by Rashkin et al. [2019] focuses on the dialogue systems' ability to empathize with the conversation partner. For this, they paired up crowdworkers on Amazon Mechanical Turk (AMT), where one of the two workers acted as Speaker and the other as the Listener. The Speaker chooses an emotion from a list of 32 options and describes a situation where she felt this emotion. The Listener is asked to respond to the Speakers by being aware of the underlying emotion (see Figure 6).

The dataset contains around 25k conversations, and each conversation consists of up to 6 turns. In contrast to the *DailyDialog* dataset, the *EmpatheticDialogues* dataset is explicitly created to convey emotions, whereas *DailyDialog* contains emotion annotations. Thus, in *EmpatheticDialogues*, all conversations are about emotions, whereas in *DailyDialog* there are neutral conversations.

### 2.6.1.4 Twitter Conversation Corpus

The Twitter Corpus by Ritter et al. [2010] contains 1.3 million conversations, which were collected using the Twitter API. In contrast to the above corpora, the Twitter corpus contains naturally occurring conversations. However, the conversations' nature is not necessarily between two persons; rather, it follows a thread of answers to a single tweet. Each conversation contains between 2 and 243 posts, where around 69% of conversations contain 2 only posts. In contrast to other corpora, the Twitter

**Input description:** list airport in ci0  
**Output  $\lambda$ -calculus:**  
`lambda $0 e ( and ( airport $0 )  
                  ( loc:t $0 ci0 ) )`

Figure 7: Example of ATIS corpus taken from Sun et al. [2019].

corpus suffers from noisy and ungrammatical utterances, complicating the texts’ processing.

## 2.6.2 Natural Language Interfaces to Databases

Here we introduce the most important corpora for NLIDB. Note that we do not use these corpora for our experiments. In chapter 4, we propose a novel methodology for creating an NLIDB corpus and a novel corpus. We introduce the following corpora to motivate the work in chapter 4.

The generation of resources for NLIDB is often time and cost-intensive since there is a need to recruit experts in SQL or SPARQL. There are two main approaches to generating the resources: either the question and the query are written by human experts, or the SQL query is randomly created based on a template and is automatically translated to an NL question, which is then paraphrased by a crowdworker. The first approach usually results in small-scale datasets, whereas the second approach usually leads to large amounts of simple queries. In the following, we review the dataset created using the aforementioned methodologies.

### 2.6.2.1 Human Annotated Datasets

The oldest dataset is the Air Travel Information System (ATIS) dataset, a dataset for task-oriented dialogue systems [Hemphill et al., 1990]. It contains dialogues consisting of long interactions between humans and a travel booking system operated by a human. The dialogues are annotated with, among other things, lambda expressions statements, which map a user query to a lambda expression (see figure 7). The dataset’s primary goal is the training of dialogue systems, where the querying of the database serves as one of many auxiliary tasks. The dataset contains 5000 user queries, which spans one database of 32 tables. The dataset contains only a tiny amount of *GROUP BY* operations and no *JOIN* operations.

Which countries in Europe have at least 3 car manufacturers?

```
SELECT T1.country_name
FROM countries AS T1 JOIN continents
AS T2 ON T1.continent = T2.cont_id
JOIN car_makers AS T3 ON
T1.country_id = T3.country
WHERE T2.continent = 'Europe'
GROUP BY T1.country_name
HAVING COUNT(*) >= 3
```

Figure 8: Example of the Spider corpus taken from Yu et al. [2018b].

Another early dataset is GeoQuery [Zelle and Mooney, 1996], which contains 877 questions about US geography that are originally annotated with Prolog. Later, Iyer et al. [2017] converted the queries to SQL. The database is rather small, containing 6 tables only. Similar to ATIS, GeoQuery does not contain any *JOIN* operations. It was constructed based on a log of questions submitted to a search website and manually annotated. Related to the GeoQuery corpus is the Restaurants corpus by Tang and Mooney [2000], which was created in the same fashion as the GeoQuery corpus. It contains only 378 questions.

The first manually created large-scale dataset is created by Yu et al. [2018b]. It contains over 10,000 questions and SQL pairs spanning over 200 databases, where each database has 5.1 tables on average. It was collected by recruiting computer science students to create question-to-SQL pairs. For each database, 20-50 pairs were created, which correspond to 5,600 data points. To enhance the size and diversity of the dataset, the questions were paraphrased by native English speakers. The SQL queries are diverse and complex (see figure 8). On average, it contains 1.19 *JOINS* per query, and 26.2% of queries contain *GROUP BY* operations.

### 2.6.2.2 Semi-automated Datasets

The automation of dataset creation is usually done by automatically creating a query and automatically generating the natural language questions. In the second step, the generated question is paraphrased by crowdworkers. There are two ways to automatically sample the query: either a set of predefined templates are filled with attributes and values from the database, or the query is sampled by applying the production rules of the underlying grammar. In the same way, the automatically

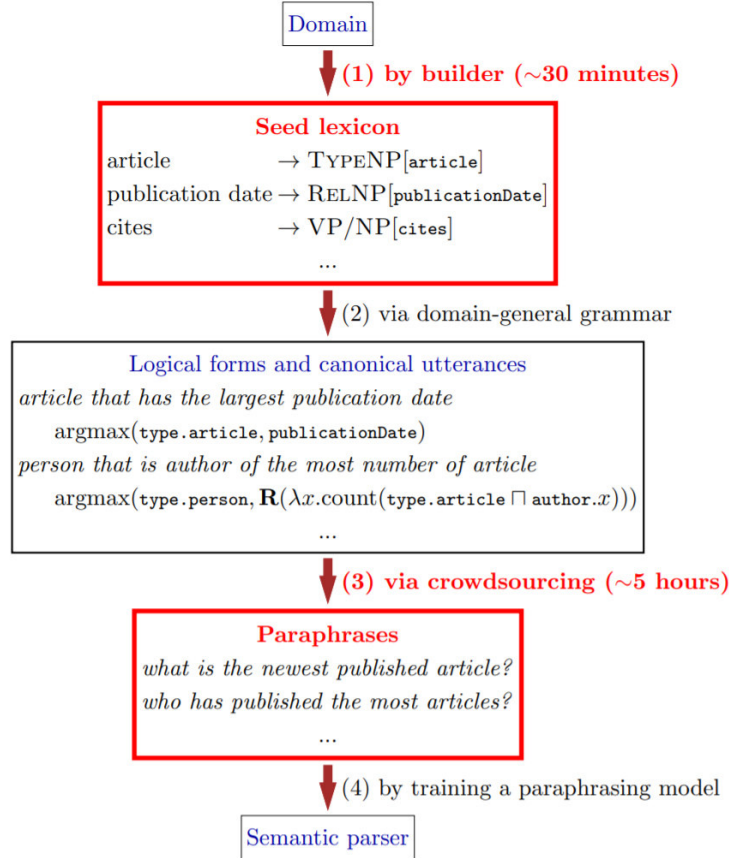


Figure 9: Process of the creation of the *OVERNIGHT* corpus taken from Wang et al. [2015].

generated natural language questions are either created following a template-based approach or by coupling the production rules of the structured question to natural language templates.

The *OVERNIGHT* corpus by Wang et al. [2015] is created by following a two-step approach of first sampling the query and the natural language question jointly from a grammar, and in a second step, the generated questions are paraphrased by crowdworkers. The queries in this dataset are represented by a logical language, which works over a database composed of sets of triples  $(e_1, p, e_2)$ , where  $e_1, e_2$  are entities (e.g., Cristiano Ronaldo, Juventus), and  $p$  is a property (e.g., playsIn). The corpus is limited to queries that work on a set of entities of a single type (unary) or on a set of entity pairs in the same property (binary). This method allows to quickly gather large amounts of data since each paraphrasing takes only around 30 seconds. The resulting dataset consists of around 27,000 examples spanning seven domains.

Zhong et al. [2018] create the *WikiSQL* corpus, which to date is the largest existing corpus with over 80,000 samples. For this, SQL queries are randomly generated; the questions are generated using templates. Similar to the *OVERNIGHT* approach, crowdworkers paraphrase the automatically generated texts. The underlying data are tables from Wikipedia. Thus, the dataset spans a large number of topics. However, the queries are limited to work over a single table.

Dubey et al. [2019] propose the *LC-QuAD 2.0* corpus, which consists of around 30,000 pairs of natural language questions and SPARQL queries. Thus, it works over knowledge graphs, more precisely DBpedia [Lehmann et al., 2015], which is a triple store containing over 400 million facts. The dataset is generated by first creating SPARQL templates, which are filled with entities and suitable predicates. Using a template-based approach, the SPARQL query is translated to a natural language question, paraphrased by crowdworkers. The dataset contains 22 different templates, around 21,000 unique entities, 1,310 unique relations. The queries also contain multi-relation queries, that is, queries that span over several relations.

# 3 Bot-Bot-Talk: A new Paradigm for the Evaluation of Conversational Dialogue Systems

Current evaluation methods for conversational dialogue systems suffer from various drawbacks. Static context-based evaluation methods (i.e., where the pair of human-written context and machine-generated response is evaluated) do not consider the multi-turn behavior of the dialogue system. On the other hand, dynamic context-based evaluations (i.e., where the context contains utterances by the dialogue system) suffer from high costs, as humans need to be recruited, who need to be trained to converse with the system properly. Furthermore, current methods are unable to compare and evaluate the relative performance of two bots directly.

In this chapter, we introduce two methods for evaluation to address the above issues. First, we introduce *Spot The Bot*, an evaluation framework based on automatically generated dialogues between two bots. This reduces the cost of evaluation since no human-bot conversations are needed. Furthermore, it allows comparing two bots directly. The methods are shown to be reproducible and efficient. Second, we introduce *AutoJudge*, an automated metric that works by first generating bot-bot-talk dialogues rated by humans. These ratings are then used to fit an automated judgment model, which correlates well to human judgments.

## 3.1 Spot The Bot

This work is an extension of Deriu et al. [2020c]. More precisely, the section 3.1.6 is novel, whereas the rest is mostly from Deriu et al. [2020c].

*Spot The Bot* is a manual evaluation framework used to rank bots by their ability to mimic human behavior. Furthermore, the framework returns a *Survival Analysis* on how many turns a bot can pass as a human. *Spot The Bot* is based on two observations:



- Conversational agents are trained on conversations between two humans (see PersonaChat, Dailydialog, and Empathetic Dialogues, see section 2.6.1). Thus, the resulting model should be evaluated based on its ability to mimic human behavior. Distinguishing between human-and bot-created utterances is a better-defined concept than concepts such as *quality* or *appropriateness*.
- Current dialogue systems are not yet human-like enough to fool humans. Thus, the longer the conversation goes on, the higher the likelihood that a bot exhibits non-human-like behavior. Hence, we compare the bots based on the ability to hide their non-humanity the longest.

### 3.1.1 Definition of Spot The Bot

*Spot The Bot* employs a tournament among chatbots to determine which performs the best at mimicking humans' conversational behavior. The tournament is based on pairwise comparisons between two bots, which lets bots talk to each other. The conversations are annotated by crowdworkers, who decide for each participant in the conversation if it is a human, a bot, or unsure. Crowdworkers are shown conversations between bots mixed with conversations between humans. Thus, there is no need to create conversations between humans and bots. Since we assume that bots are not yet strong enough to fool humans for more than a couple of turns, we add a time component. That is, we show the conversation at different points in time. The bot that wins the most comparisons and is spotted only in later turns ranks higher in the tournament.

Figure 10 shows the annotation process. The conversation is shown after 2, 3, and 5 exchanges. At each of these segments, the annotator states for each conversation participant if they think it is a bot, a human, or unsure. Each annotation is interpreted as a game, in which the bot that can fool the human wins.

**Formal Setup.** Assume a pool of  $b$  bots  $\{B_1, \dots, B_b\}$ , which is to be ranked. For each pair of bots, a set of conversations is sampled by letting the bots talk to each other, where  $S_{ij}$  denotes the set of conversations between bots  $B_i$  and  $B_j$ , where  $i \neq j$ . Each conversation is defined as a sequence of exchanges  $e_0, \dots, e_N$ , where each exchange consists of two turns:  $e_i = \{t_0^{e_i}, t_1^{e_i}\}$ , one for each participant.

**Segmentation.** The more exchanges there are in a conversation, the more likely it is that a bot gets recognized as such. Thus, we show different segments of the conversation to the crowdworkers. A segment is defined as the first  $k$  exchanges of the dialogue:  $S_{ij}^k = e_0, \dots, e_k$ . Thus, an annotator only sees the first  $k$  exchanges of the conversation. Each segment of the same conversation is rated by a different

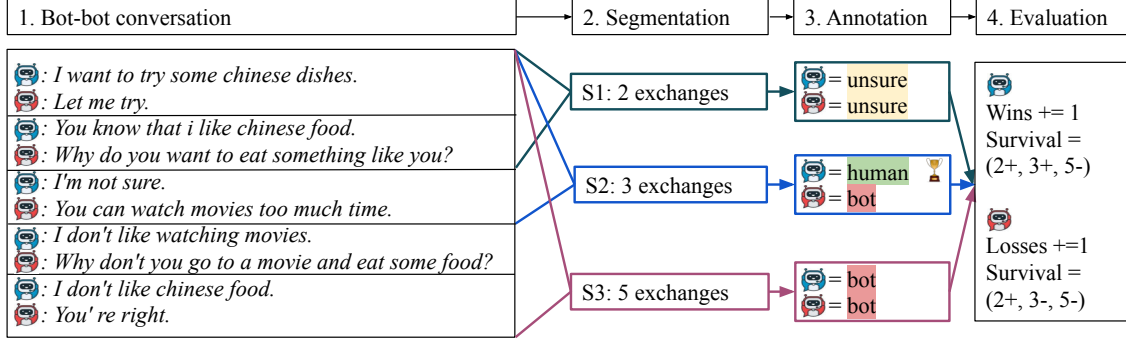


Figure 10: Figure taken from Deriu et al. [2020c]. Example for the Spot the Bot pipeline. First, conversations between two bots are sampled. The conversations are presented to the crowdworkers at different points in time (e.g., after 2, 3, or 5 exchanges. The crowdworkers annotate for each participant in the conversation if they think it is a bot or human. The winner of the conversation is then determined.)

annotator to avoid that one annotator sees parts of the same conversation multiple times, which would bias the rating. We choose different segment lengths since we cannot know a priori whose length is sufficient for the different bots to be recognized. Alternatively, we tested letting crowdworkers decide themselves when they are sure about the nature of a participant. For this, they could turn-by-turn decide to either trigger the decision process or look at the next turn. However, this approach led to undesirable behavior, as crowdworkers showed a large variety in their behavior. For instance, some always decided after a single exchange or first opened all the exchanges and then decided.

**Human Conversations.** We add conversations among humans to the pool of conversations that are to be rated. The human conversations are sampled from the training set used to train the dialogue systems in the respective domain. The results of the annotations of the human dialogues establish an upper bound for the evaluation. Also, they are meant to prevent annotators from concluding that all participants are bots. We investigated if annotators realize that conversations are either between bots or humans by looking at ratios of conversations where both participants are labeled identically but found no evidence that this happens more often than by chance.

**Annotation.** The annotation procedure works in two steps: First, the annotators decide for each participant in a conversation segment if it is a bot or a human. Second, to correlate the outcome to various characteristics of a bot, the framework allows rating specific features (e.g., fluency or appropriateness). The framework

then measures these features' influence on the survival time of the bots, which serves as an explainability component. The annotation tool is depicted in Figure 11.

**Annotation Features.** We chose three features: sensibleness, specificity [Adiwardana et al., 2020], and fluency. The first two are shown to capture the core conversational behavior of answering sensibly and not with contradictory or false statements while being specific to the conversations' given context. More precisely, sensibleness is defined to capture whether the utterance is sensible to the current dialogue. Stated with Grice's maxims: whether the utterance clear (i.e., easy to understand, unambiguous, and brief), logically coherent, or factually correct. For instance, if the bot states that it is currently working as a nurse, but a couple of turns later claims to be without a job, this is factually incorrect and not sensible. However, a bot can be sensible by just stating unspecific and universal answers (e.g., "I don't know"). Thus, specificity measures if the bots' response is specific to the context. That is, the answer is only valid in the specific context. For instance, if the human states that she likes to go shopping then, "Ok." would be an unspecific answer (although sensible); however, "Me too, I especially like online shopping!" is a specific answer.

The third feature states if the utterances are grammatically correct and fluent. The features are rated by preference ranking. That is the annotator states which of the two participants performed better regarding the features.

### 3.1.2 Output of Spot The Bot

There are two outputs of *Spot The Bot*, first a ranking of the bots based on the pairwise comparisons, and second, a *Survival Analysis*, which for each bot returns the survival probability at each point in time (i.e., after each exchange). Thus, there are two views on the results: a relative view (ranking) and an absolute view (Survival Analysis). Furthermore, the Survival Analysis allows for investigating which features correlate the most with survival, which allows for a more fine-grained analysis of the bots.

#### 3.1.2.1 Ranking of Chatbots

We define a win function for the annotations of the pairwise, direct conversations between two bots. The outputs of the win function are aggregated to determine the overall winner of the tournament.

Required Worker ID  
Annotator1

Conversation Progress  
73.333%

Package Progress  
0/15

Annotated Packages  
0/3

Entity: 0  
do you know familiar persons ?

Entity: 1  
hey , 2 people , and you ?

Entity: 0  
i 'm not .

Entity: 1  
i have 5 sons , and you ?

Entity: 0  
i do not drink , i drink a lot of mountain biking

Entity: 1  
okay , are you male ?

Entity: 0  
i am a female , i am a female .

Entity: 1  
i coach little league basketball

Entity: 0  
i like to go mountain bike and mountain biking

Entity: 1  
what is your favourite colour ?

Entity: 0  
i like blue , reminds me of summer time

Entity: 1  
i like orange

Is Entity 0

HUMAN BOT UNSURE

SUBMIT

Is Entity 1

HUMAN BOT UNSURE

SUBMIT

(a) Human/Bot Decision

Required Worker ID  
Annotator1

Conversation Progress  
73.333%

Package Progress  
0/15

Annotated Packages  
0/3

Entity: 0  
do you know familiar persons ?

Entity: 1  
hey , 2 people , and you ?

Entity: 0  
i 'm not .

Entity: 1  
i have 5 sons , and you ?

Entity: 0  
i do not drink , i drink a lot of mountain biking

Entity: 1  
okay , are you male ?

Entity: 0  
i am a female , i am a female .

Entity: 1  
i coach little league basketball

Entity: 0  
i like to go mountain bike and mountain biking

Entity: 1  
what is your favourite colour ?

Entity: 0  
i like blue , reminds me of summer time

Entity: 1  
i like orange

Who performed better in:

Fluency: Entity 0 Entity 1 No Difference

Sensibleness: Entity 0 Entity 1 No Difference

Specificity: Entity 0 Entity 1 No Difference

SUBMIT

(b) Features Decision

Figure 11: Taken from Deriu et al. [2020c]. There are two steps to the annotation process. First (a), the Human/Bot decision, where the crowdworkers are presented with the first n exchanges of a conversation and need to decide if it is a bot or not. Second (b), the crowdworkers annotate their preference with regard to certain features.

**Win Function.** Each annotation at each segment length  $S_{ij}^k = e_0, \dots, e_k$  of a conversation constitutes the result of one annotation applied by one crowdworker, individually labeling each of the two participants as either *bot*, *human*, or *unsure*. The winner of segment  $S_{ij}^k$  under a crowdworker’s annotation is determined by the following ordering of the labels:  $human > unsure > bot$ . That is, if bot  $B_i$  is assigned the label *human* and bot  $B_j$  has label *bot* or *unsure*,  $B_i$  has won the segment. Similar to Bojar et al. [2013], we define a win rate of  $B_i$  against  $B_j$  to aggregate the wins from all segments of all annotations stemming from conversations between bots  $B_i$  and  $B_j$ , as:

$$\frac{\text{WINS}(B_i, B_j)}{\text{WINS}(B_i, B_j) + \text{WINS}(B_j, B_i)} \quad (3.1)$$

where  $\text{WINS}(B_i, B_j)$  denotes the number of times that  $B_i$  wins against  $B_j$ .

**Ranking.** To create the ranking, we follow the approach by Dušek et al. [2018], where the ranking is generated by the TrueSkill algorithm [Herbrich et al., 2006] based on the win rate, and significant differences in performance are determined by bootstrap sampling. The result is a ranked set of clusters, where each cluster is composed of participants that do not have a significant difference in performance. More formally, let  $C = (c_1, \dots, c_n)$  be the set of all annotated pairwise conversations, then bootstrap sampling works by subsampling  $m < n$  samples, let  $\tilde{C} = (c_1^*, \dots, c_m^*)$  be the subsampled set of annotated conversations. A ranking is then computed based on the subsampled set of annotations (e.g., using TrueSkill or just using the win rate). This process is then repeated multiple  $s = 10,000$  times; thus, there are potentially  $s$  different rankings. For each bot, the range of positions is computed in which it appears at least 95% of times. For instance, if a bot appears 9,000 times in first place and 501 times in second place, it is assigned the range  $1^{st}$  to  $2^{nd}$  place. In order to determine the significance of the positions, a clustering based on ranges is computed. A cluster is composed of ranges, which overlap, which means two clusters contain ranges, which have no overlap. For instance, the ranges in one cluster range from 1 to 4 and the ranges in the second cluster range from 5 to 7.

### 3.1.2.2 Survival Analysis

While pair-wise win rates are well-suited to provide a *relative* ranking among a pool of bots, it does not serve as an *absolute* evaluation of a single bot’s ability to disguise as a human. The conversations’ segmentation introduces a time component, which we leverage to investigate our intuition that bots are more likely to reveal

themselves in longer conversations. In our evaluation, a bot that can disguise in long conversations is most successful. Thus, we complement our evaluation with *Survival Analysis*.

Survival Analysis estimates probabilities for the occurrence of an event at different points in time. It has a long history in the medical domain, where it is used to estimate the effectiveness of different treatments [Li and Ma, 2013]. It is applied in engineering disciplines to estimate the time to failure of machine components [Eyal et al., 2014]. In our case, we are interested in the time corresponding to the number of exchanges until a dialogue system is spotted. Besides, Survival Analysis allows us to correlate finer-grained characteristics to the survival probability, which allows us to inspect which of the annotated features impact a bot’s survival.

We interpret the annotation data as such: the *spotted* event occurred if the system was annotated as “bot” and it *survived* if it was annotated as “unsure” or “human”. Let  $k$  be the number of exchanges in the annotated conversation segment, meaning each dialog system produced  $k$  outputs. If the dialog system was not spotted, we know it survived for at least  $k$  exchanges. This is a so-called right-censored data point. If the dialogue system was spotted as such, we cannot tell the exact number of exchanges it took for an annotator to spot it, meaning it could have taken less than  $k$  exchanges. We thus record that the spotting event happened in the interval  $(0, k]$ , a so-called interval-censored event.

From this data, we can get non-parametric estimates of the survival function of the different systems per domain [Turnbull, 1974]. To check whether these differences are significant, we apply a generalized log-rank test [Zhao and Sun, 2004]. We use the *Cox Proportional Hazards Model* [Cox, 1972] to study the influence of the features on the time before the systems are spotted.<sup>1</sup>

### 3.1.3 Experiments: Application of Spot The Bot

**Domains.** We apply *Spot The Bot* to three widely used domains for conversational dialogue systems: Dailydialog [Li et al., 2017], Empathetic Dialogues [Rashkin et al., 2019], and PersonaChat [Zhang et al., 2018] (see Table3). For each domain<sup>2</sup>, we prepared a pool of bots to be ranked and analyzed. For each pair of bots, we sampled  $|S_{ij}| = 45$  conversations. We seed the conversations by using the first exchange of a conversation in the test set, which is sampled at random. Although there exists a probability that the bots resample parts of a conversation, we did not find evi-

---

<sup>1</sup>We use the *icenReg* R package [Anderson-Bergman, 2017], which allows us to fit a Cox model to our interval-censored data.

<sup>2</sup>See details in Section 2.6.1.

DOMAIN NAME	#DIALOGUES	AVG. EXCHANGES	$ B $	SEGMENTS
DAILYDIALOG	13118	3.74	4	2,3,5
EMPATHETIC DIALOGUES	25000	1.65	5	1,2,3
PERSONACHAT	10907	7.85	6	2,3,5

Table 3: Overview of the domains. The number of dialogues, the average number of exchanges in the training data, the bot pool size, and the segment lengths are show. Taken from Deriu et al. [2020c].

dence of this happening. In fact, only 2% of all sampled conversations contain an exchange, which can be found in the training material.

**Annotation.** For the annotation task, we recruited paid crowdworkers from Amazon Mechanical Turk (AMT). To avoid that the results are biased towards a few crowdworkers’ performances, we designed a Human Intelligence Task as a batch of 20 conversations, and each worker was only allowed to work on three batches. We designed the batches so that two segments of the same conversations never appear in the same batch, and each batch contains different segments of different conversations. Two different annotators annotate each sample <sup>3</sup>.

**Segmentation.** The segment lengths are based on the lengths of the dialogues in a domain. Since we add human conversations of the training set to be rated, the sampled dialogues should adhere to their lengths. PersonaChat and Dailydialog have long conversations; thus, we used segments of 2, 3, and 5 exchanges. The Empathetic Dialogue domain has short dialogues; thus, we used segment lengths of 1, 2, and 3 exchanges.

**Dialogue Systems.** For each domain, we prepared a pool of dialogue systems to be ranked. If applicable, we reused existing systems. To assess the performance of *Spot The Bot* regarding weak models, we trained a small sequence-to-sequence model (DR) for only 3 epochs, which returns mostly general answers. For the Dailydialog domain, we trained all bots in the pool using ParlAI [Miller et al., 2017] as there were no pre-trained models available. To leverage the recently developed language models, we fine-tune a GPT-2 (GPT) model [Radford et al., 2019], and

---

<sup>3</sup>Note that the restrictions of having only two annotations per sample were due to limitations in the budget.

a BERT-Rank (BR) model. Additionally, we train a sequence-to-sequence model (S2) with attention to compare the language models to previous state-of-the-art approaches. Together with the DR model, the pool consists of  $b = 4$  systems. For the Empathetic Dialogues, we prepared the same pool of models as in Dailydialog. Since the recently developed Blender model [Roller et al., 2020] is trained on the Empathetic Dialogue dataset as well, we add the pre-trained version to the pool (BL). For the PersonaChat domain, we mostly reuse the openly available systems of the ConvAI2 challenge [Dinan et al., 2020], namely, Lost in Conversation<sup>4</sup> (LC) and Huggingface<sup>5</sup> (HF), which were the top-rated dialogue systems in the ConvAI2 challenge [Dinan et al., 2020], as well as KVMemNN (KV), which served as a strong baseline. We also add the Blender model, which is also trained in this domain. In order to have more retrieval-based systems, we train a BertRank (BR) model. Together with the DR model, the pool consists of six different dialogue systems.

### 3.1.3.1 Ranking Results

Table 4 gives an overview of each pair of bots’ win rates and their ranking ranges. The Chi-square test computes the significance. For each domain, most pairwise win-rates are significant.

As expected, DR performs worst in all three domains due to its repetitive nature, which is exposed over the course of a dialogue. In all cases, the win-rate of DR lies between 12 – 19%. In the Dailydialog and the Empathetic Dialogues domains, the GPT2 and the BR models perform equally, i.e., they end up in the same cluster. For instance, in Dailydialog, GPT2 and BR lie in the range-cluster 1<sup>st</sup> to 2<sup>nd</sup> place. In both domains, systems using pre-trained language models outperform the S2 model, which is learned from scratch, aligning with related findings’ expectations. The BL model outperforms all other models in both the PersonaChat and Empathetic Dialogues domains, which is in line with the results presented by the authors of the Blender model [Roller et al., 2020]. Furthermore, the LC model is ranked very highly. This corresponds to the findings of the ConvAI2 challenge [Dinan et al., 2020]. However, in *Spot The Bot*, the KV is ranked much higher than the HF model, which is not in line with the ConvAI2 evaluation.



Dailydialog							
	GPT	BR	S2	DR		WR	RANGE
GPT2	-	<b>0.67</b>	<b>0.77</b>	<b>0.93</b>		0.79	(1,1)
BERT RANK	<b>0.33</b>	-	<b>0.79</b>	<b>0.83</b>		0.65	(1,2)
SEQ2SEQ	<b>0.23</b>	<b>0.21</b>	-	<b>0.74</b>		0.39	(3,3)
SMALL SEQ2SEQ	<b>0.07</b>	<b>0.17</b>	<b>0.26</b>	-		0.16	(4,4)
Empathetic Dialogues							
	BL	BR	GPT	S2	DR		RANGE
BLENDER	-	<b>0.82</b>	<b>0.83</b>	<b>0.9</b>	<b>0.94</b>	0.87	(1,1)
BERT RANK	<b>0.18</b>	-	0.51	<b>0.77</b>	<b>0.93</b>	0.59	(2,3)
GPT2	<b>0.17</b>	0.49	-	<b>0.61</b>	<b>0.73</b>	0.50	(2,3)
SEQ2SEQ	<b>0.10</b>	<b>0.23</b>	<b>0.39</b>	-	<b>0.63</b>	0.33	(4,4)
SMALL SEQ2SE	<b>0.06</b>	<b>0.07</b>	<b>0.27</b>	<b>0.37</b>	-	0.19	(5,5)
PersonaChat							
	BL	LC	KV	HF	BR	DR	RANGE
BLENDER	-	0.56	<b>0.68</b>	<b>0.72</b>	<b>0.84</b>	<b>0.95</b>	(1-1)
LOST-IN-CONVO	0.44	-	0.54	<b>0.72</b>	<b>0.75</b>	<b>0.89</b>	(2-3)
KVMEMNN	<b>0.32</b>	0.46	-	<b>0.77</b>	<b>0.74</b>	<b>0.91</b>	(2-3)
HUGGINGFACE	<b>0.28</b>	<b>0.28</b>	<b>0.23</b>	-	<b>0.63</b>	<b>0.89</b>	(4-4)
BERT RANK	<b>0.16</b>	<b>0.25</b>	<b>0.26</b>	<b>0.37</b>	-	<b>0.75</b>	(5-5)
SMALL SEQ2SE	<b>0.05</b>	<b>0.11</b>	<b>0.09</b>	<b>0.11</b>	<b>0.25</b>	-	(6-6)

Table 4: Win rates (WR) for each pair of systems for each of the three domains. The bold entries denote significance ( $p < 0.05$ ) computed with Chi-square test. The ranking ranges are computed using bootstrap sampling. Table taken from Deriu et al. [2020c].

### 3.1.3.2 Survival Analysis Results

Figure 12 shows the survival functions for the three domains. The survival rates produce the same rankings as those from pairwise win rates reported in Table 4, except for the Empathetic Dialogues domain, where GPT and BR switch places. Importantly, the distinction between these two is not significant in any of the rankings. Further non-significant differences within the Survival Analysis are S2 and DR in the Empathetic Dialogues domain, BR and S2 in the Dailydialog domain, and

<sup>4</sup>[https://github.com/atselesousov/transformer\\_chatbot](https://github.com/atselesousov/transformer_chatbot)

<sup>5</sup><https://github.com/huggingface/transfer-learning-conv-ai>

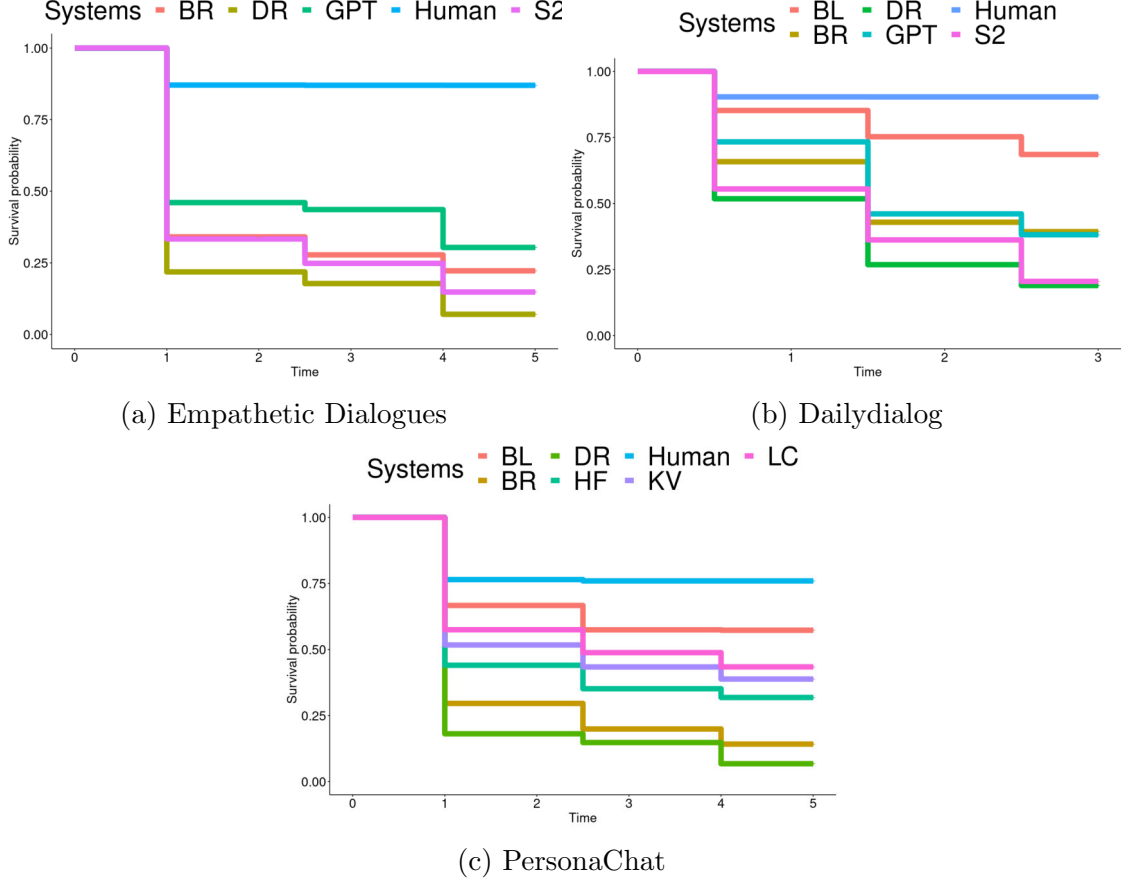


Figure 12: Taken from Deriu et al. [2020c]. Survival function per system estimated for each domain. Best viewed in color.

LC and KV for the PersonaChat domain. All other pairwise comparisons of survival curves are significant with  $p < 0.05$  after correction for multiple comparisons.

**Feature Influence.** For each of the three features – fluency, specificity, and sensibleness – annotators have to specify whether one participant performed better, the same, or worse than the other. We encode this information as 1, 0, and  $-1$  respectively and fit a Cox proportional hazards model [Cox, 1972] for every system independently with the features as covariates.

The numerical entries in Table 5 refer to the per-feature win-rate of each bot, which is computed analogously to Equation 3.1 using the feature annotations directly. Bold entries in Table 5 show which features significantly influence the system being spotted. All significant effects go in the intuitive direction, meaning that a higher feature value leads to longer survival. For example, for the DR model, the fluency

Dailydialog			
	Fluency	Specificity	Sensibleness
GPT2	<b>0.69</b>	0.55	<b>0.77</b>
Bert Rank	0.77	<b>0.78</b>	<b>0.62</b>
Seq2Seq	0.31	0.52	<b>0.41</b>
Small Seq2Seq	<b>0.23</b>	0.15	<b>0.20</b>
Empathetic Dialogues			
	Fluency	Specificity	Sensibleness
Blender	0.84	0.79	<b>0.84</b>
GPT2	<b>0.51</b>	0.42	<b>0.49</b>
Bert Rank	<b>0.60</b>	0.65	<b>0.56</b>
Seq2Seq	<b>0.33</b>	0.47	<b>0.39</b>
Small Seq2Seq	<b>0.21</b>	0.17	<b>0.21</b>
PersonaChat			
	Fluency	Specificity	Sensibleness
Blender	<b>0.73</b>	0.74	<b>0.73</b>
Lost-In-Convo	0.56	0.54	<b>0.62</b>
KVmemNN	<b>0.61</b>	0.63	<b>0.58</b>
Huggingface	<b>0.46</b>	<b>0.</b>	<b>0.47</b>
Bert Rank	0.48	0.44	<b>0.43</b>
Small Seq2Seq	<b>0.16</b>	0.19	<b>0.16</b>

Table 5: Per feature win-rate of the different systems over all domains. Bold numbers indicate that the feature has a significant influence on system survival according to a Cox model.

feature is significant across all three domains, and together with its low fluency win rate, we can deduce that it is often spotted due to its low fluency. Sensibleness seems to be an important feature across the board, meaning that, in general, bots can be spotted due to inappropriate, nonsensical answers or hide if they respond appropriately. Interestingly, specificity seems to be mostly unimportant, which could be due to either the bots not being noticeably unspecific or an irrelevant feature for the chosen domains.

label	bot ↓	human ↑	unsure
<i>human</i>	<i>0.33</i>	<i>0.84</i>	<i>0.15</i>
Blender	0.38	0.65	0.14
Lost-In-Convo	0.60	0.52	0.10
GPT2	0.65	0.48	0.15
Huggingface	0.70	0.41	0.10
KVmemNN	0.64	0.49	0.08
Bert Rank	0.74	0.39	0.15
Small Seq2Seq	0.85	0.29	0.17

Table 6: Inter-annotator agreement on labels.

### 3.1.4 Discussion of Spot The Bots’ Advantages

#### 3.1.4.1 Inter-Annotator Agreement

The robustness of the evaluation of chatbots is often hampered by inter-annotator agreement (IAA) [Gandhe and Traum, 2016]. Measuring and reporting IAA is not yet a standard practice in evaluating chatbots [Amidei et al., 2019b], and producing annotations with high IAA on open-domain conversations is prone to be impeded by subjective interpretation of feature definitions and idiosyncratic annotator behavior [Bishop and Herron, 2015].

In our setting, annotator disagreement on a bot’s human-like behavior can be interpreted as a *feature* of a bot’s performance: A bot that manages to fool one of two annotators into believing it is human can be said to have performed better than a bot that does not manage to fool any annotator.

To analyze the annotator agreement in this light, we calculate per bot and label the percentage of cases where both annotators annotate the label if one of them does. Given three labels (*human*, *bot*, *unsure*), the chance for random agreement is 0.33. The results averaged over all investigated domains and segment lengths per bot are shown in Table 6.<sup>6</sup>

The results confirm that the bots that rank high based on win rates and in the survival analysis (BL, GPT, LC) obtain the highest agreement on the *human* label and lowest agreement on the *bot* label. Conversely, the DR system obtains the highest agreement when identified as bots and lowest when perceived as a human.

---

<sup>6</sup>We also analyzed agreement per segment length and domain but found no significant difference to averaging agreement over domains and segment lengths.

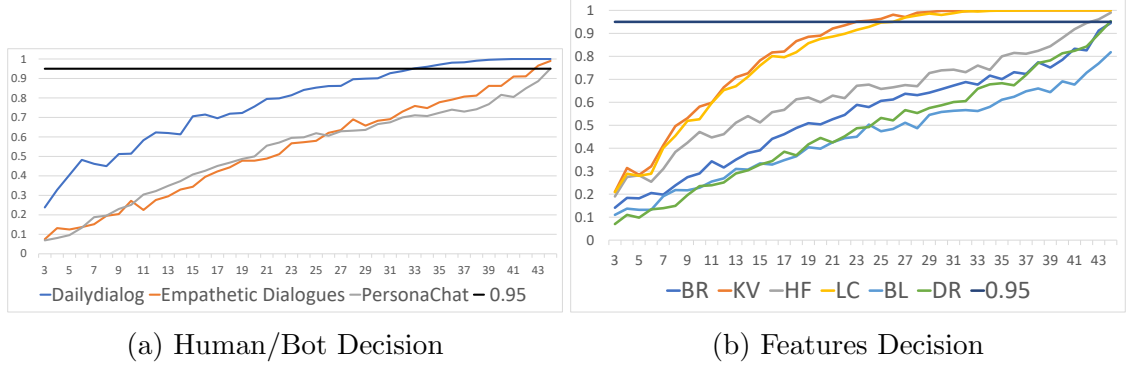


Figure 13: Taken from Deriu et al. [2020c]. Ranking stability experiments. The x-axis denotes the number of pairwise conversations between two bots. The y-axis denotes the rate at which the same ranking is achieved across 1000 repetitions. The horizontal line denotes the 95% mark. In the lower Figure, we show the experiments for the PersonaChat domain when leaving one system out. Best viewed in color.

This analysis suggests that our experiments’ results do not stem from a random agreement between the annotators, i.e., the annotations of the best and worst-performing systems show agreement distinctly higher than chance regarding the respective labels.

### 3.1.4.2 Reproducibility and Robustness

One key requirement for an evaluation procedure is that repeated executions of the procedure result in the same outcome, i.e., that the procedure is reproducible. We measure how many pairwise conversations between two bots are needed to guarantee a stable ranking. That is, what is the lower bound to  $|S_{ij}|$  so that the ranking is stable. For each  $|S_{ij}| \in \{3...45\}$ , we randomly sample  $|S_{ij}|$  conversation for each pair and compute the ranking. We repeat this subsampling procedure 1000 times and measure the minimum  $|S_{ij}|$  that guarantees the same ranking in at least 95% of cases.

Figure 13a shows for each  $|S_{ij}| \in \{3...45\}$  the proportion of times in which the most frequent ranking occurred. For the Dailydialog domain,  $|S_{ij}| = 33$  pairwise conversations are enough to guarantee a stable ranking. In the other two domains, this value is reached with over 40 pairwise dialogues.

A more in-depth analysis reveals that ranking stability depends on the significance of pairwise comparisons. For instance, in the PersonaChat domain, the KV and

DOMAIN	Annotation Time (Sec)	Time per Conversation (Sec)
DAILYDIALOG	26	153
EMPATHETIC DIALOGUES	18	136
PERSONACHAT	24	238

Table 7: Overview of time efficiency in Seconds. Spot The Bot annotation versus creating human-bot conversations.

LC systems are not significantly different, which leads to two different rankings depending on the subsampling: in the first, KV and LC are in the same cluster, and in the second, LC and KV are in separate clusters, with LC being on top. Thus, removing either of them from the pool would yield a more stable ranking. To investigate this further, we applied a leave-one-out stability analysis. More precisely, we applied the analysis on  $B \setminus \{sys_i\}$ , where  $sys_i \in B$ . Figure 13b shows the result of the leave-one-out stability analysis. When leaving one between LC or KV out, stability is achieved with 25 pairwise dialogues. When removing one of the other systems, stability is reached with at least 40 dialogues. Thus, the number of pairwise bot-bot chats needed for *Spot the Bot* evaluation depends on the pool of bots to be evaluated. For a novel domain or a different set of bots, the number of dialogues to be sampled  $|S_{ij}|$  is to be determined empirically using the above method.

### 3.1.4.3 Time Efficiency

Usually, evaluation methods are costly and take up a long time, slow down the development cycle of dialogue systems. *Spot The Bot* brings down the cost and time effort compared to other methods. In Table 7 the mean time per annotation is displayed. The time is tracked by the annotation tool, measuring the time needed to annotate one sample. For the Dailydialog and PersonaChat domain, the average annotation time is at around 25 seconds. For the Empathetic Dialogues, it is at 18 seconds, which is due to the shorter dialogues. We compare this to the time to create conversations between humans and bots. We recruited three dialogue system experts from our lab to interact with the systems. The experts are experienced in the development of dialogue systems, fluent in English and experienced in the usage of conversational dialogue systems. Each expert created 5 conversations with each system. The average times do not take into account the time needed to instruct the experts. For the Dailydialog and Empathetic Dialogues domains, it takes over

DOMAIN	#ANN	AVG. CORR	AVG. HUM. CORR.	< 50%
DAILY DIALOG	33	77%	86%	9.1%
EMPATHETIC DIALOGUE	32	63%	92%	7.5%
PERSONA CHAT	40	69%	77%	22.8%

Table 8: Overview of the annotator performance. The number of annotators (#Ann), the average correctness score (AVG. CORR), the average correctness score for the human-human conversations (AVG. HUM. CORR.), and the percentage of annotators that have a correctness score below 50% (< 50%).

2 Minutes per conversation.

For PersonaChat, the time increased to almost 4 minutes. Similarly to our experts, the average time for a human-bot conversation in the wild evaluation of the ConvAI2 challenge<sup>7</sup> also lies at 4 minutes<sup>8</sup>. Considering the 100 dialogues per system used in ConvAI, the evaluation time would be 2,000 minutes per system. In Spot the Bot, 40 annotations times 24 seconds mean 16 minutes per pair of systems. Assuming a comparison between 5 systems, an approach based on human-bot annotations such as ConvAI would require 20 thousand minutes, while Spot the Bot would do with 0,16 thousand minutes<sup>9</sup>.

#### 3.1.4.4 Stability against weak Annotators

One drawback of Likert-scale-based evaluation methods is that many annotations need to be removed due to unreliable annotators Lowe et al. [2017a]. *Spot The Bot* shows that it is stable against weak annotators. Since we can measure how often the annotators correctly classify a participant, we can rate an annotator’s quality. A random annotator would receive a correctness rate of 50%. Table 8 shows an overview of the annotators for each domain.

For each domain, 32 to 40 different annotators worked on the task. The average

<sup>7</sup><http://convai.io/data/>

<sup>8</sup>We consider only conversations that have at least 10 turns, which is comparable to the setting of our experts.

<sup>9</sup>The amount of time needed by ConvAI grows linearly with the number of systems, while Spot the Bot (and ACUTE-EVAL) would grow quadratically. A pool of five systems seems reasonable for a research team, but even for larger pools (up to 51 systems) *Spot the Bot* is still more efficient.

correctness score is significantly higher than random (i.e., 50%). For instance, for the Dailydialog domain, the correctness rate lies at 77%. For PersonaChat, this rate is lower because stronger models are in the pool, which fool the annotators more often. For Empathetic Dialogues, this rate is even lower, which lies in the fact that the segments are shorted, which leads to annotators having less information for their decision.

The average correctness on humans measures how well the annotators are correctly labeling humans as such. While the rates for Dailydialog and Empathetic Dialogues are high, this rate is the lowest for PersonaChat with 77%, that is, in 23% of cases, the humans were mistaken for bots. One possible explanation is that this is due to the rather unnatural setting in which the training data was accumulated, resulting in artificial sounding dialogues.

For the Dailydialog and Empathetic Dialog domain, the rate of annotators, which achieved a rate below 50%, was below 10% of all annotators. For the PersonaChat domain, the rate is higher because stronger dialogue systems were in the pool of bots. The average correctness scores for correctly predicting humans is high for all domains. Hence, *Spot The Bot* proves to be stable against annotators with low scores.

When removing all annotators with scores below 75%, the rankings remain stable. Only the significance scores decrease as a large number of dialogues gets removed. This lies in contrast to the gathering of conversations between humans and bots, which must be strictly supervised. For instance, the dialogues gathered in the wild evaluation of the ConvAI2 challenge were not usable. We applied *Spot The Bot* on these conversations, and the humans were rated as bots in 45% of the cases.

### 3.1.5 Qualitative Analysis

Table 9 shows two conversations between two bots from the PersonaChat domain. The first conversation is between DR (the undertrained sequence-to-sequence model) and KV (the Key-Value Memory Neural Network). It shows one common type of errors made by neural dialogue systems: the repetitions within a single utterance, e.g., "i am a student, but i am a student". Such an utterance is characteristic of neural generative models. This example also shows the effects of the interaction, i.e., how one bots' performance influences the other. The response selected by KV, "no. i have one sister. ..." does not make sense in this context. The bot answered a question, which was never posed.



Example 1: PersonaChat		
DS	Utterance	Annotation
DR	i am doing great , how are you ?	Human
KV	doing well , all things considered . . have a lot on my mind these days . .	Human
DR	what do you do for a living ?	Human
KV	i am in marketing . mostly social media management .	Human
DR	i am a student , but i am a student .	Bot
KV	no . i have one sister . she sings first grade songs . you live at home ?	Bot
DR	that is cool . i am a student .	Bot
KV	ah , i can already tell i am gonna like you . i have that power .	Human
Example 2: PersonaChat		
DS	Utterance	Annotation
KV	awesome ! i want to get a dog once i move out .	Human
HF	i love to dance and eat meat , and pigs !	Human
KV	that is nice . no do not dance but i like to sing	Human
HF	i don't dance but i do love country music	Human
KV	i love to dance as well	Bot
HF	i don't dance but i 'm a vegan	Human
KV	i love to dance everyday	Bot

Table 9: Examples of annotations for the PersonaChat domain. The double lines denote the segments. That is the first segment consists of the first 4 turns, the second segment consists of the first 6 turns, and the last segment consists of the full sampled dialogue.

The second example is a conversation between KV and HF (a GPT-2 based model). It shows a more subtle type of error, namely, not being consistent. In the 3<sup>rd</sup> turn, KV states that it does not dance, only to state two turns later that it loves to dance. Being consistent is still an unsolved problem for neural chatbots.

In both cases, the crowdworkers annotated the bots correctly as bots at the turns in which the failures occurred. In the first conversation, the KV bot is labeled as a human again in the last turn. However, this is because each segment is annotated by a different crowdworker, which leads to a bot "recovering" from being spotted.

### 3.1.6 Automation of Spot The Bot

One obvious question to address is the automation of Spot The Bot. Is it possible to train a model that learns to distinguish between humans and bots, which can be used to run Spot The Bot automatically? Unfortunately, there is no trivial way to accomplish this. There are fundamental problems that need to be solved. Here, we first introduce the experiments, which all yield a negative result, then discuss possible reasons for the observed behavior.

We recreate the Spot The Bot setting for the experiments, composed of automatically sampled dialogues between the bots in the aforementioned pool. We performed the experiments on the PersonaChat domain since the bot pool is the most diverse there. We sampled 500 conversations for each pair of bots, yielding 7.5k dialogues between bots. We sampled an equal number of human conversations from the dialogue corpus.

Although there are various modeling approaches, we found that they all yield the same sort of results. To showcase the problems, we opt to model the task as a sequence labeling task, where the model needs to label each turn as either bot or human.

**Sequence Labeling.** Here, we employ a hierarchical neural network architecture, where each turn is encoded using RoBERTa Liu et al. [2019b], and then, we apply a recurrent neural network over the sequence of hidden states. More formally, let  $D = t_1^1, t_2^1, \dots, t_1^n, t_2^n$  be the sequence of turns comprising the dialogue  $D$ , and each turn consists of a sequence of tokens:  $t_{1,2}^i = w_1, \dots, w_{m_i}$ . RoBERTa returns the hidden state for each turn as a high dimensional vector  $h_{1,2}^i \in \mathbb{R}^N$ . Thus, the dialogue is represented as a sequence of hidden states:  $H^D = h_1^1, h_2^1, \dots, h_1^n, h_2^n$ , which is used as input to the RNN to predict the labels for each turn. We refer to this RNN as the *dialogue-level RNN*. Let  $v_{1,2}^i$  denote the hidden state of the *dialogue-level RNN*. At each time step, the RNN predicts the next label, which is either human or bot.

$$v_1^i = RNN(h_1^i, v_2^{i-1}) \quad (3.2)$$

$$o_1^i = \sigma(Wv_1^i + b) \quad (3.3)$$

The value  $o_1^i \in \{0, 1\}$  is used to determine the winner of the conversation. If  $o_1^i > o_2^i + \lambda$ , that is if the confidence of bot 1 is larger than the confidence of bot 2 with a margin of at least  $\lambda$ , then bot 1 wins segment  $i$ . If the values  $o_1^i$  and  $o_2^i$  lie within a margin of  $\lambda$ , then the segment is counted as a tie. We use  $\lambda = 0.05$  as threshold.

PersonaChat								
	BR	KV	LC	BL	HF	DR	WR	RANGE
BERT RANK	-	0.56	<b>0.66</b>	<b>0.61</b>	<b>0.80</b>	<b>0.85</b>	0.70	(1-2)
KVMEMNN	0.43	-	<b>0.57</b>	<b>0.67</b>	<b>0.76</b>	<b>0.81</b>	0.65	(1-3)
LOST-IN-CONVO	<b>0.34</b>	<b>0.43</b>	-	0.54	<b>0.67</b>	<b>0.65</b>	0.53	(2-3)
BLENDER	<b>0.39</b>	<b>0.33</b>	0.46	-	<b>0.60</b>	<b>0.83</b>	0.52	(4-5)
HUGGINGFACE	<b>0.20</b>	<b>0.24</b>	<b>0.33</b>	<b>0.40</b>	-	0.53	0.35	(4-5)
SMALL SEQ2SEQ	<b>0.15</b>	<b>0.19</b>	<b>0.35</b>	<b>0.17</b>	0.47	-	0.27	(5-6)

Table 10: Win rates (WR) for each pair of systems for each of the three domains according to the automated scores.

There are two considerations to be made. First, since we only have human-to-human conversations or bot-to-bot conversations in the dataset, we need to avoid that the neural network uses this information. For this, we create two logistic regression layers, one for each participant:

$$o_1^i = \sigma(W_1 v_1^i + b_1) \quad (3.4)$$

$$o_2^i = \sigma(W_2 v_1^i + b_2) \quad (3.5)$$

The second consideration is that a bi-directional RNN should not be used since looking into the future should be avoided. Experiments showed that bi-directional RNNs lead to almost perfect performance on the task so that all the "matches" end up in draws.

We perform 5-fold cross-validation. The model achieves an average accuracy score of 98.4%, which almost solves the task of distinguishing between humans and bots perfectly<sup>10</sup>. However, this is not the intended outcome since we are interested in a model, which behaves similarly to human judges. Table 8 shows that human annotators achieve an accuracy score of around 70%, which is far lower than the models' score. We apply bootstrap sampling to create the ranking as described in section 3.1.2.1.

Table 10 shows the win rates and the ranking, which results from the automated Spot The Bot version. In 69% of cases, the conversation is rated as a draw. Most strikingly, the BL model is ranked only at fourth place and is put in the same

---

<sup>10</sup>Note that the model is trained on the ground-truth, i.e., if the turn was uttered by a human or a bot. The model is not trained on human-annotated data, i.e., if the an annotator thinks a turn is uttered by a human or a bot.

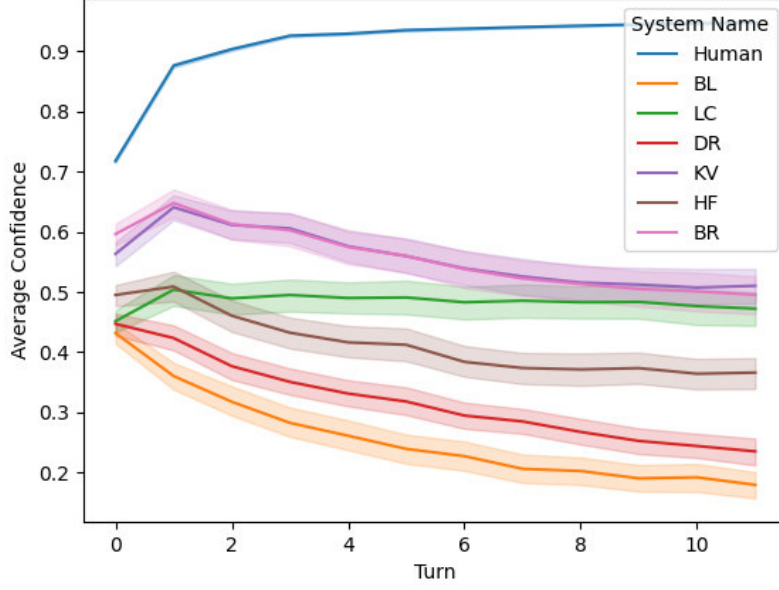


Figure 14: The averaged confidence score  $o_{1,2}^i$  over the number of turns. A score of 1.0 indicates that the model rates the interlocutor with full confidence as human, and a score of 0.0 indicates that the model rates the interlocutor as a bot with full confidence.

cluster as DR, with contradicts the human ranking. Interestingly, we note that BR and KV, which are both utterance-selection models, are placed at the top of the ranking. Utterance-selection models retrieve human-written utterances making them inherently more human-like on the surface level. This result indicates that the automated Spot The Bot model only learns features on the surface level and disregards other dialogue -specific features, such as consistency or the bots’ ability to maintain the context.

To better understand the trained model, we analyze the confidence scores for the dialogue systems after each turn. Figure 14 displays the average confidence scores for each dialogue system, i.e., the scores  $o_{1,2}^i$  averaged over all conversations. First, we note that for all bots, the score decreases over the number of turns. For example, DR has an average score of 0.45 after the first turn and a score of 0.28 after 11 turns. The score for humans increases over the turns from 0.71 to 0.96. Most strikingly, the BL system, which performed best in Spot The Bot, has the lowest confidence. This fact highlights one of the fundamental issues with the approach, namely, learning human behavior. BL is trained on different dialog corpora next to PersonaChat and exhibits a different dialogue behavior as the ones in PersonaChat. For instance, BL

asks more questions. This leads to BL being classified as a bot since it follows a different distribution than the human dialogues. On the other hand, KV and BR have the highest scores among all the bots due to them returning human-written utterances.

## 3.2 AutoJudge

This section is published in Deriu and Cieliebak [2019].

In the previous section, we introduced bot-to-bot talk for generating dialogues automatically and used these dialogues to be rated by crowdworkers to create a robust method for human evaluation for conversational dialogue systems. However, the approach is not easily automated. In this section, we introduce *AutoJudge*, an evaluation method that is trained on a set of labeled dialogue turns. In contrast to other trained metrics, which are trained on a static context, *AutoJudge* is trained on dynamic contexts generated by self-talk<sup>11</sup>. That is, by letting a bot talk to itself. The self-talk conversations are then rated by crowdworkers on a turn basis on a Likert scale from 1 to 5 and used to fit a regression model, which we refer to as *AutoJudge*. Finally, *AutoJudge* is applied in various ways to improve the bot’s performance. We show that *AutoJudge* is able to achieve state-of-the-art correlations to human judgments. We show the limitations of *AutoJudge* by applying it as a reward for reinforcement learning, where the weaknesses of *AutoJudge* are exploited.

### 3.2.1 The Process to build AutoJudge

Figure 15 shows an overview of the *AutoJudge* framework. Given a pool of bots  $B$ , it works in three phases. In phase one, each bot applies self-talk to generate dialogues of a certain length automatically. Each dialogue consists of a sequence of turns  $D = t_0, \dots, t_N$ . In the second phase, each turn of the automatically generated dialogues is annotated with regard to the appropriateness of the turn given the context. The annotated data is then used as training material for a regression model. That is, each turn is labeled:  $D = (t_0, y_0), \dots, (t_N, y_N)$ , where  $y_i \in \{1, \dots, 5\}$ . In the last phase, a regression model is fitted on the labeled data, which is then applied to evaluate and improve the dialogue systems. More precisely, we apply *AutoJudge* in two ways to improve the dialogue systems. First, as a ranking module where the output of various dialogue systems are re-ranked based on the regression scores (bottom left part of figure 15). Second, as a reward for reinforcement learning (bottom right part of figure 15). This method is trained to be applicable in situations where there is no gold response available.

---

<sup>11</sup>We regard self-talk as a special case of bot-bot-talk where the same instance of a bot talks to itself.

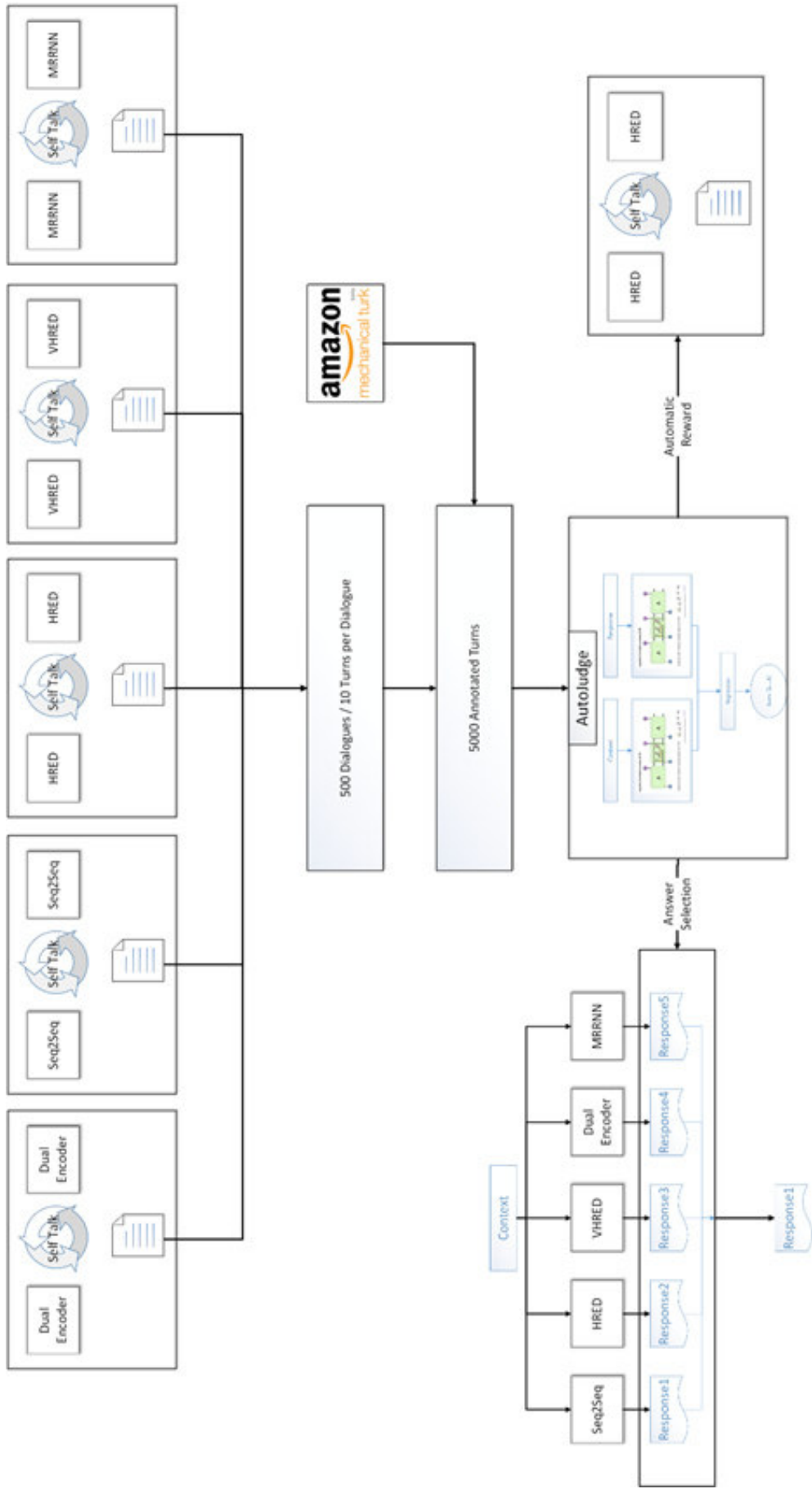


Figure 15: Overview of AutoJudge. See text for details.

### 3.2.1.1 Dialogue Systems

We relied on the following state-of-the-art conversational dialogue systems for our experiments, which we introduced in more detail in section 2.2.1:

**Seq2Seq.** As proposed by Vinyals and Le [2015], the Sequence-to-Sequence model consists of an encoder and a decoder. Both modules are based on Long Short-Term Memory cells (LSTM) [Hochreiter and Schmidhuber, 1997], where the encoder consumes the last utterance and produces a hidden representation, which is passed as an initial state to the decoder to condition the generation process.

**HRED.** The Hierarchical Recurrent Encoder-Decoder (HRED) model proposed by Serban et al. [2016] enhances the Seq2Seq model by a hierarchical encoding procedure. Here, the context-turns are encoded by first encoding each turn separately and then applying a recurrent encoder over the turns' hidden states. The decoding procedure is conditioned on the hidden state produced by the context encoder.

**VHRED.** The Hierarchical Latent Variable Encoder-Decoder model (VHRED) by Serban et al. [2017c] enhances the aforementioned HRED model by introducing a stochastic latent variable at the utterance level. This stochastic variable aims to inject variability at the utterance level, which increases the variety of responses a model generates.

**MrRNN.** The Multi-resolution Recurrent Neural Network (MrRNN) [Serban et al., 2017a] enhances the HRED model by introducing an abstraction layer. More precisely, the dialogue is modeled by processing the inputs and outputs at various levels of abstractions (e.g., at the level of meaning-bearing words and the usual word level).

**DE.** The Dual Encoder (DE) [Lowe et al., 2015] is a selection-based model, which differs from the generation-based approaches of the aforementioned models. The DE encodes both the context and a candidate response (using the same encoder as the VHRED model) and then classifies if the candidate is a valid response to the given context.

**Model Training.** We used a bidirectional LSTM to encode the turns and a unidirectional LSTM for both the context encoder and decoder for all turns. We specify the number of units for the LSTMs to 500, 1000, 1000 for the turn-encoder, context-encoder, and decoder, respectively. We use the pre-trained 300-dimensional FastText embeddings [Bojanowski et al., 2017], which we refine during the training. In order to avoid too large vocabularies, we limit the vocabulary size to 20k distinct tokens. The set of tokens are determined by selecting the 20k most frequent tokens from the training set. The generative models are trained to minimize the reconstruction error. For the VHRED and MrRNN, we refer to the original papers for the loss function formulation. The Dual Encoder is trained to minimize a contrastive loss



function  $\log \sigma(c^T r_{True}) + \sum_{n \in N} \log \sigma(-c^T r_n)$ , where  $c$  is the context encoding,  $r_{True}$  is the correct response encoding and  $N$  is a set of negative samples. For each training sample, we sampled 10 negative examples uniformly at random from the training set.

### 3.2.1.2 Turn-Level Annotation

We apply *self-talk* to generate dialogues automatically. For this, we sample 100 different contexts randomly from a set of unseen contexts and let the dialogue system generate a dialogue starting from this context, which consists of 10 turns each. For the annotation process, we use Amazon Mechanical Turk (AMT)<sup>11</sup> and follow the procedure outlined by Lowe et al. [2017a] (see section 2.3.2.1), i.e., the judges rated the *overall quality* of each turn on a scale from 1 (low quality) to 5 (high quality). Three different judges annotate each turn. We required the AMT workers to be from an English-speaking country (USA, UK, Ireland, or Australia) to ensure that they are native speakers since the generated messages are highly colloquial and make heavy slang usage. We paid 15 cents for each annotation, where we assumed that each annotation takes between 60 to 90 seconds. For selecting the final turn-label, we apply the MACE procedure [Hovy et al., 2013], which learns confidence scores for the annotators. Our final dataset consists of 500 annotated dialogues, which amounts to 5000 annotated pairs of contexts and responses.

### 3.2.1.3 Training AutoJudge

Similarly to the *ADEM* procedure proposed by Lowe et al. [2017a], we train a regression model on the annotated data. For this, we use the pre-trained context and response encoder from the VHRED model. Unlike *ADEM*, our dialogues are generated automatically. Thus, we do not have access to a gold-standard response. For this reason, we use the following scoring function:  $score(c, r) = (c^T M r - \alpha) / \beta$  where  $M \in \mathbb{R}^{d \times d}$  is a learned similarity matrix,  $\alpha, \beta$  are scalar constants, and  $c, r$  are the context and response embeddings respectively. The model is optimized to minimize the mean squared error between the predicted ratings and human judgments.

---

<sup>11</sup><https://www.mturk.com/>

### 3.2.1.4 Improving Dialogue Systems with AutoJudge

Since *AutoJudge* is fully automated, we apply it to improve the existing dialogue systems. For this, we implemented the following two applications: as a reward for reinforcement learning (RL) and as ranking candidate utterances.

**3.2.1.4.1 Ranking.** Given a list of responses from the five aforementioned dialogue systems for a given context, *AutoJudge* ranks them by their predicted score. In our experiments, we use the dialogue systems, which we trained for the self-talk experiment, i.e., we rank the outputs of the five aforementioned dialogue systems. Thus, the *AutoJudge* serves as a meta-selection module.

**3.2.1.4.2 Reinforcement Learning Reward.** We apply the predicted ratings as a reward in the RL framework. For this, we apply the Policy Gradient formulation, similar to Li et al. [2016b]. However, we applied the Advantage Actor-Critic (A2C) formulation [Sutton and Barto, 2018], which is more robust against the high variability in the vanilla formulation of Policy Gradient.

$$\nabla J_{RL}(\theta) = \sum_i \nabla \log p(r_i | c_i) \times \sum_i A(r_i, c_i) \quad (3.6)$$

, where  $r_i$  and  $c_i$  are the response and context in the  $i^{th}$  turn, and  $\sum_i \log p(r_i | c_i)$  is the reconstruction error. The term  $A(r_i, c_i)$  is the advantage function, which is defined as:  $A(r_i, c_i) = R(r_i, c_i) + \gamma V(c_{i+1}) - V(c_i)$ , where  $R(r_i, c_i)$  is the reward function for the action given the current state, that is, the reward for utterance  $r_i$  given the dialogue context  $c_i$ . Note that  $R(r_i, c_i)$  is modelled by *AutoJudge*. The term  $V(c_i)$  is the value function, which computes the expected return starting from the context  $c_i$ . Note that  $V(c_i)$  is modelled as a neural network, which takes the current context and estimates the expected return. The algorithm below shows how the policy network (i.e. the dialogue system) is trained using the A2C formulation.

## 3.2.2 Results and Discussion

In our experiments we use the Twitter Dialogue Corpus by Ritter et al. [2011]<sup>11</sup>. The Twitter Dialogue Corpus provides social interactions. Since social interactions are easier to understand for a more significant part of the population than the technical

---

<sup>11</sup>We use the IDs provided by Serban et al. [2017c], which can be found here: [www.iulianserban.com/Files/TweetIDs.zip](http://www.iulianserban.com/Files/TweetIDs.zip)

---

**Algorithm 1:** Batch Advantage Actor-Critic Algorithm

---

```

while training do
    sample  $\{c_i, r_i\}$  using self-talk (use AutoJudge to rate the turns);
    fit  $V(c_i)$  to sampled reward sums;
    compute  $A(r_i, c_i) = R(r_i, c_i) + \gamma V(c_{i+1}) - V(c_i)$  ;
     $\theta = \theta + \alpha \nabla J_{RL}(\theta)$ 
end

```

---

	Pearson Corr	Spearman's Rho	MAE
CONVO SPLIT	0.573	0.577	0.928
SYSTEM SPLIT	0.544	0.53	0.984

Table 11: Average correlations between the judgements predicted by AutoJudge and the human judgement. CONVO SPLIT denotes the cross-validation split according to the contexts and SYSTEM SPLIT denotes the cross-validation split according to the dialogue system. Table taken from Deriu and Cieliebak [2019].

contents of the Ubuntu Corpus, we believe them to be a reasonable basis for being annotated via crowdsourcing.

### 3.2.2.1 Data Aggregation

The turn-level ratings provide us with 5000 annotated pairs of context and responses. The distribution over the labels is balanced (i.e., each class is represented between 19% and 21% of the cases). However, the human judges' agreement scores are low: the median pairwise Spearman correlation between two judges is only 0.403. Furthermore, the MACE procedure reports on the confidence score (between 0 and 1) of single judges, which is used as a basis for selecting the final label. The average confidence is at only 0.15. We assume that these problems stem from the high degree of subjectivity of the problem.

### 3.2.2.2 AutoJudge

We train *AutoJudge* using k-fold cross-validation. There are two ways of splitting the data into folds to ensure that all turns of the same dialogue are in the same fold. First, we group the 100 contexts into 10 folds, and each fold consists of 50 dialogues (i.e., 10 contexts times the number of dialogue systems), which we denote

as CONVO SPLIT. The second option is to split the data according to the system which created the conversation, which evaluates the performance of *AutoJuge* in rating dialogues of unseen dialogue systems. We denote this as SYSTEM SPLIT. In Table 11, we report the average Pearson correlation, Spearman’s rho and mean absolute error (MAE) over all folds for the *conversation split* and the *system split*. With moderate correlations of 0.573 on the dialogue level, we get results comparable to Lowe et al. [2017a], where ADEM achieves a Pearson correlation of 0.436. Note that we cannot directly compare our results to the BLEU score and ADEM since these base their predictions on gold standards, which we do not have in our setting. An interesting result is the *System Split*, i.e., that our approach maintains a high correlation (0.544) with the ratings of a dialogue system when removing the data of that system from the training, which is not the case in Lowe et al. [2017a] where the correlation for a different system dropped significantly.

Furthermore, we evaluate the correlation on the system level. For this, we compute the average ratings over all turns from each dialogue system. We compare the average ratings from the judges with the average predicted ratings over 4000 randomly sampled dialogues. Table 12 shows these ratings. Although the absolute scores do not correspond between the humans and *AutoJudge*, the Pearson correlation is at 0.793.

Lastly, we present a qualitative analysis of the judgments computed by *AutoJudge*. In Table 14 three randomly sampled self-talk dialogues are shown with the ratings computed by *AutoJudge* next to the human ratings. Generally, we see that *AutoJudge* overestimates the scores where the human rating is below 3. For instance, in turn 1, in dialogue #3, the interlocutor thanks the other for the following him on Twitter and asks about the phone battery status. However, in turn 2, the other participant does not answer the questions but talks about something completely different. The human judges rated this turn badly (i.e., 2 out of 5 points). *AutoJudge*, on the other hand, rates the dialogue as 3.55 out of 5 points, which would be a moderate response. However, *AutoJudge* is able to pick up on the deterioration of the self-talk dialogues over the course of different turns. This is shown by the fact that humans give lower scores for later turns as the self-talk dialogue deteriorates, *AutoJudge* displays the same behavior.

### 3.2.2.3 Answer Selection.

To evaluate the improvements achieved by the ranking method, we sample a disjoint set of 100 new contexts and apply *self-talk* to generate conversations. Then, we use

Systems	Avg. Turn Ratings	
	Human	Predicted
SEQ2SEQ	3.512	3.699
HRED	2.956	3.638
VHRED	3.392	3.676
MRRNN	2.792	3.626
DUAL ENCODER	2.612	3.378

Table 12: Human ratings compared to the predicted ratings on the system level. One rating is the average rating over all turns of one dialogue system. The Pearson correlation between the average human rating and the average predicted rating is 0.793 and a Spearman correlation is 1. Table taken from Deriu and Cieliebak [2019].

Systems	Dialogue Level Rating
SEQ2SEQ	<b>3.31</b>
HRED	2.78
VHRED	3.20
MRRNN	2.37
DUAL ENCODER	2.02
RE-RANKING	<b>3.47</b>

Table 13: Human judgements on the dialogue level for each dialogue system. For this, each dialogue system (the five base-systems and the re-ranking system) generate 100 dialogues using self-talk, which human judges rated on the dialogue level. Here we see the average ratings for each system. Table taken from Deriu and Cieliebak [2019].

AMT to let humans judge the automatically generated conversations on the dialogue level (i.e., a rating for the entire dialogue instead of turn-based ratings). We compare the performance of the five base dialogue systems to the performance of the ranking strategy. Table 13 shows the average scores for each dialogue system. Our results show that the *ranking* approach raises the score to 3.47, which is 0.16 points higher than the best base-system (i.e., SEQ2SEQ).

### 3.2.2.4 Reinforcement Learning.

We use the pre-trained HRED system as the initial policy to apply the reinforcement learning procedure. We sample a batch of 100 self-talk dialogues at once in order

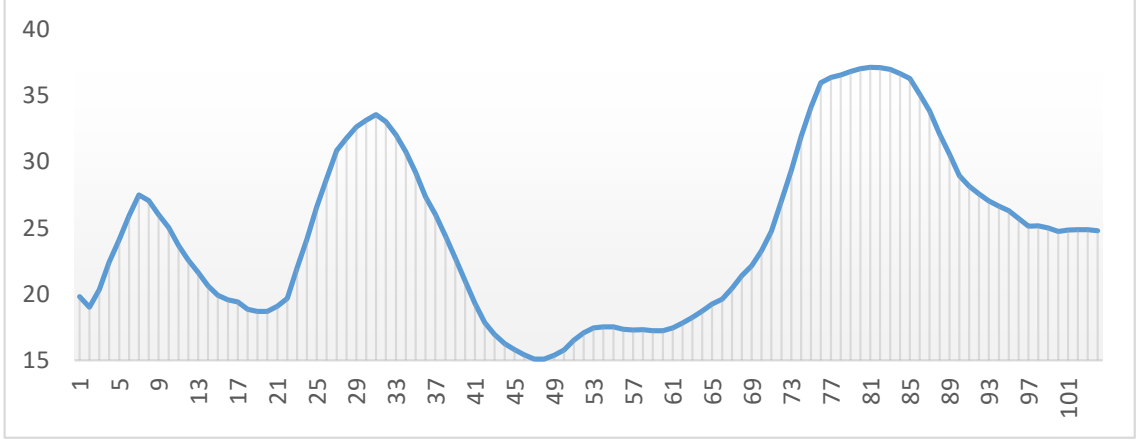


Figure 16: Reinforcement Learning rewards over time. The x-axis shows the number of episodes (i.e., number of self-talk dialogues) and the y-axis shows the cumulative reward, that is, the summed reward over all turns of the dialogue.

to reduce the variance. When we apply *AutoJudge* as a reward, the dialogue policy deteriorated. Although the return increases over time (from 21.74 to 37.41 over 80 episodes), the dialogues which the policy generates are often incoherent or completely useless (see Figure 16). In most cases, the dialogues converge to a sequence of empty turns rated highly by *AutoJudge*.

This behavior is most likely since *AutoJudge* can easily be gamed, a well-known problem for trained metrics[Sai et al., 2019]. Thus, the policy finds ways to optimize its responses to those that maximize the reward, which is not necessarily the types of responses that are regarded as appropriate (e.g., empty responses, repeating responses, convergence to a single universal response). It is not clear how *AutoJudge* can be stabilized. We experimented by adding constructed negative samples manually to train *AutoJudge* to increase the robustness. However, the learned policy converged to different types of bad dialogues. Another stabilizing procedure is to regularize the reward by using a language model (i.e., BERT), which computes the probability of an utterance  $p_{lm}(r_i)$ . Thus the reward is the multiplication of the language model probability and the *AutoJudge* score. This combination leads to single-word utterances, which are highly rated by the language model.

#	#Turn	Utterance	AutoJudge	Human
1	1	@user ur prolly tired now , arent u ? wanna sleep , don't cha ? (:	4.67	5.00
	2	@user atm jus chillin nd chattin nd listening to music :P haha xxx	3.70	2.00
	3	@user x factor tickets came so im siva happy dancing :) might do some fanfic xx wbu ? xx	3.26	1.00
	4	@user goood :) do u want to be in my 1d fanbook ? xx	3.07	2.00
	5	@user : o gosh ppl these days :/ ... ok tell me everything 2moz p . s send me the link so i can send it to et :) xx dont stress it :)	2.89	1.00
	6	@user yer im fine thanks xxx	3.18	3.00
2	0	if i could i would hurt you ... a lot ! ! i'm so sick of you blaming me for everything that goes wrong ! !	4.86	5.00
	1	@user awh ... is everything ok sweetie ?	4.72	5.00
	2	@user thanks ... it is ... i just can't wait to see how it is ...	3.72	3.00
	3	@user wait , what ?	3.51	5.00
	4	@user i have a new phone . i have to get it .	2.54	3.00
	5	@user how much do you pay ?	2.20	3.00
	6	@user pay attention	2.65	1.00
3	0	#hashtag @user @user @user @user @user @user @user @user	4.47	5.00
	1	@user thanks for the follow . do you have at bat on your phone yet ?	4.95	5.00
	2	@user there's a lot going on ... but i think is good to say what u think here and at the boards too . specially there !	3.55	2.00
	3	@user pretty much just as well as the current macbook pros do ...	2.42	1.00
	4	@user how well does it run photoshop / illustrator ?	2.67	2.00
	5	@user this is like sixth sense . can i have your games consoles ?	3.00	1.00
	6	@user probably , but i don't know if it'd be appropriate in this case . i try only to use our funds for things i know i need to know .	2.15	1.00

Table 14: Randomly sampled output. The conversation is sampled at random and *AutoJudge* rates each turn. Table taken from Deriu and Cieliebak [2019].

### 3.3 Summary

This chapter introduces using bot-to-bot talk as a new paradigm to evaluate conversational dialogue systems. We applied this to two evaluation methods.

First, we introduced Spot The Bot, where bot-to-bot talk automatically creates conversations between pairs of bots. Humans annotate the conversations to evaluate the human-likeness of the interlocutors. Based on these annotations, a ranking is created based on which dialogue system could convince the human judges of its intelligence the most. The results show that Spot The Bot is more efficient than letting humans talk to bots while yielding robust results. Spot The Bot is stable against weak annotators as it does not suffer from low annotation agreements, a problem that plagues other evaluation methods. By applying a survival analysis, we can get insight into which bots can survive longer and correlate the survival to more fine-grained features (e.g., fluency).

The second application of bot-to-bot talk is on *AutoJudge*. Here, self-talk is applied to generate conversations automatically, which are annotated by crowdworkers regarding the quality of the turns. The annotated data is used to train a regression model that learns to emulate the human judges. While the correlation scores to human judgments are comparable to other trained metrics, the results show that applying *AutoJudge* as a reward to improve the system is not yet possible.



## 4 Evaluation for Natural Language Interfaces to Databases

In the last chapter, our discussion focused on evaluating conversational dialogue systems. The challenge lay in defining protocols for handling the unstructured nature of these types of dialogues (i.e., chitchat dialogues). In contrast, this chapter focuses on question-answering systems, more precisely on Natural Language Interfaces to Databases (NLIDB). NLIDB systems take as input a question in natural language, generate a structured query, run the query on a database, and return the result to the user. Only questions about the contents of a database are valid, thus, limiting the scope to factoid questions. The facts are stored in an SQL database. One difference to conversational dialogue systems is that NLIDB systems are more easily evaluated. They are either evaluated using the correctness of their answers or by comparing the generated query with the reference query. Thus, our investigation is not interested in novel metrics for evaluating an NLIDB system. Here, we focus on the evaluation after deployment.

This chapter describes three contributions.

- In section 4.1, we introduce a novel annotation procedure alongside a novel dataset called *OTTA*. It introduces a novel representation of structured queries, which we call operation trees (OT). This representation has two main advantages. First, it is easier to visualize and understand than plain SQL, facilitating the annotation procedure. Second, it is easier to write a question for a given OT than to invent a question and write the corresponding OT. This section extends the work published in Deriu et al. [2020a].
- In section 4.2, we introduce a novel method to automatically evaluate NLIDB systems by back-translating the generated query to natural language and applying textual semantic similarity to compare the user input to the back-translated query.
- In section 4.3, we introduce a post-processing step to re-rank the beam-search hypothesis based on back-translation and semantic textual similarity, which

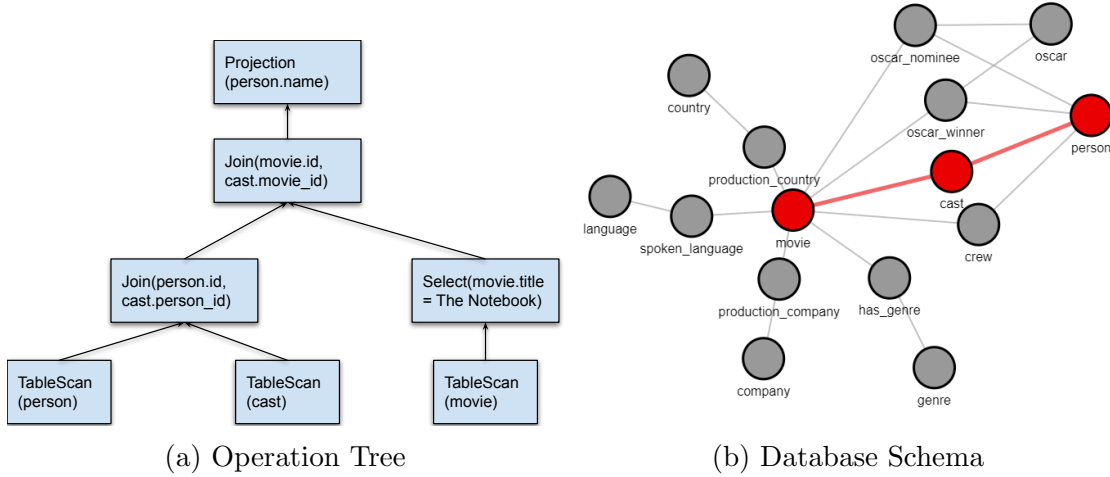


Figure 17: Taken from Deriu et al. [2020a]. (a) Example of an Operation Tree (OT) for the query "Who starred in 'The Notebook'?" (b) The corresponding database schema. Best viewed in color.

improves the performance of the NLIDB system.

## 4.1 OTTA - A corpus for NLIDB

In this section, we introduce a novel method to construct a corpus for training NLIDB systems efficiently. For this, we introduce the aforementioned operation trees (OTs). They are closely related to the logical execution plan in a relational database system. They allow for inverting the annotation procedure. We sample a new OT from a context-free grammar at random and let annotators write the corresponding question that the OT answers. This approach lies in contrast to the traditional approach of letting SQL experts invent a question and write the SQL query. Our approach triples the annotation speed compared to the traditional approach while maintaining the same complexity and variety. We use this annotation approach to create OTTA, a corpus of operation trees for five databases. Finally, we introduce *GrammarNet*, our baseline model, which we use for the remainder of this section.

### 4.1.1 Operation Trees

This section introduces Operation Trees (OT), which we use as representations for structured queries. The usage of intermediate representations for SQL queries has

been shown to aid neural networks in learning the task of translating natural language to structured queries more efficiently [Guo et al., 2019]. They usually work by translating the SQL queries of the dataset into an abstract syntax tree. In contrast, OTs are directly created as such. We leverage the ability to visualize the query as a tree to collect data more efficiently. Furthermore, it allows creating a hard alignment between the tokens of the natural language and the operations in the OT. An OT comprises a sequence of atomic operations that are easily mapped to a database query language such as SQL or SPARQL to retrieve the proper result. While the current set of operations follows the ones that map to SQL queries (e.g. *TableScan*, *Join*, *Selection*, etc.), more types of operations are conceivable. The operations are represented as a binary tree, where the children’s outputs are used as input for the parent node. For instance, in figure 17(a) the *Join* node uses as input the results of two *TableScan* operations.

**Example.** Assume that we have a database about movies that we want to query in natural language. In Figure 17, an example of an OT is depicted for the question ”Who starred in ’The Notebook’?”. In order to answer this question, the tables *person* and *movie* are selected, then the table *movie* is filtered by movie title *The Notebook*. In the next step, the tables are joined via the bridge-table *cast*. Finally, the *person.name* column is extracted.

**Token Assignment.** We enhance these OTs by associating a reasonable subset of tokens from the natural language question to each operation in the tree. For instance, the token ”starred” could be associated with the *Join* operation, as this operation implies that an actor starred in a movie (see Figure 18), whereas the tokens ”How many” could be associated with the *Count* operation. This mapping between tokens and operations will help train machine learning algorithms to generate OTs from natural language questions with better quality automatically.

**Definition.** More formally, the OTs follow a predefined context-free grammar. The set of operations includes major operations from the relational algebra with specific extensions in the current state. The full grammar is shown in Figure 19.

The OTs can be used to represent queries for any entity-relationship data paradigm. For instance, in SQL databases, the entity types are the tables, the attributes are the columns, and the relationships are represented as bridge-tables. Similar mappings are possible for other paradigms.

**Properties.** The OTs have several features:

- *Question Types:* There are different types of questions that can be asked. For instance, 1) yes/no questions (*IsEmpty*), 2) questions about a list of items

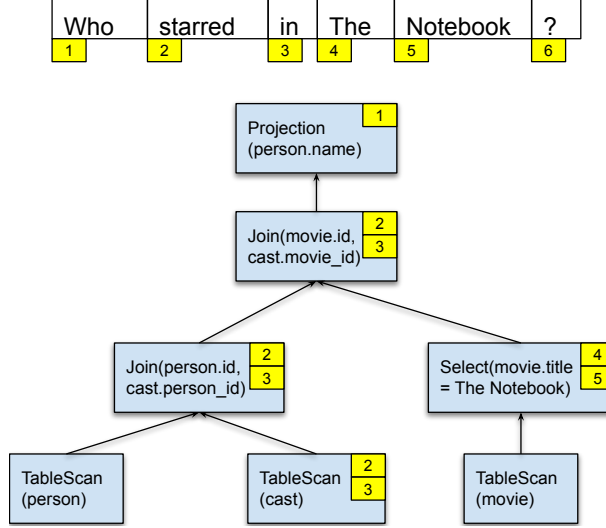


Figure 18: Example of the token assignment. Each token is assigned to zero or more operations, and each operation can be aligned to multiple tokens. For instance, the "starred in" tokens are assigned to the operations, which define the relation between movies and persons. Namely, *TableScan(cast)* defines the relation of who starred in a movie, as well as the *Join* operations, which execute the relation between movies and persons.

$S ::= done(R) \mid isEmpty(R) \mid sum(T,A) \mid average(T,A) \mid count(R)$   
 $R ::= projection(T, A)$   
 $T ::= tableScan(TN) \mid selection(T, A, OP, V) \mid min(T, A) \mid max(T, A) \mid$   
 $distinct(T) \mid join(T, T, A, A) \mid union(T, T, A, A) \mid$   
 $intersection(T, T, A, A) \mid difference(T, T, A, A) \mid averageBy(T, A) \mid$   
 $sumBy(T, A) \mid countBy(T, A)$   
 $TN ::= \text{table name}$   
 $A ::= \text{attributes}$   
 $OP ::= < \mid > \mid <= \mid >= \mid == \mid !=$   
 $V ::= \text{values}$

Figure 19: The set of production rules for the context-free grammar of the operation trees, where *table name* denotes the set of all entity types in the database, *attributes* denotes the set of all attributes of entity types, and *values* denotes the set of all entries in the database. The non-terminal symbols **S**, **T**, and **R** denote the start-symbol, intermediate tables, and result tables respectively. Figure taken from Deriu et al. [2020a].

(*Projection* followed by *Done*), 3) questions about the cardinality of a result set (*Count*), and 4) questions about an aggregation (*Sum*, *Avg*, etc.).

- *Result Types*: The type of results is defined by the entity types in the result

set. For instance, a question can ask about the list of directors that satisfy certain constraints (e.g., all directors born in France). In this case, the result type would be the *person* type.

- *Constraints*: The constraints represent the filters that are applied to the attributes of the entities. For instance, "All directors born in France" sets a constraint on the *birth\_place* attribute.
- *Entity Types*: They define which entity types are involved in the query. The selected entity types are combined, usually via a *Join* operation. For instance, in Figure 17 the entity types are *movie* and *person*, which are combined with the table *cast*.
- *Aggregation Types*: They define reduction operations, which are applied to the data. This includes *Min/Max operations* on an attribute, *Set operations* on two sets of relations, and *Group By operations*.

**Complexity.** In order to categorize the OTs, we define a complexity score similar to Yu et al. [2018b], which is based on the number of nodes in the tree. This follows the intuition that queries with more operations are harder for the NLIDB system to learn and also harder for the annotators to understand. The more *Joins*, *Group By* operations, *Aggregations* or *Selections* are in the query, the higher the score. Like Yu et al. [2018b], we define four categories: *Easy*, *Medium*, *Hard*, and *Extra Hard*. More precisely, we compute a complexity score for each OT based on a heuristic. That is:

$$\begin{aligned} \text{SCORE(OT)} = & \# \text{JOINS} + (0.5 * \# \text{SELECTIONS}) + \# \text{GROUPBY} \\ & + \# \text{SET OPERATIONS} + (0.5 * \# \text{AGGREGATIONS}) \end{aligned}$$

For instance, the OT in Figure 17 contains two *Join* operations and one *Selection*, thus, its score is 2.5 points. The categories are defined based on score ranges: the *Easy* OTs have scores  $\leq 1.0$ , which are trees that contain no *Join* operations, and only one or two *Selections*. The *Medium* OTs have scores between 1.0 and 2.5 points, which are OTs that contain one or two *Join* operations and one or two *Selection*, or no *Join* operations but a *GroupBy* operation. The *Hard* OTs are those with scores between 2.5 and 3.5 points, which are OTs with multiple *Join* operations and multiple *Selections*, they often contain *Aggregations* or *GroupBy* operations. Everything above 3.5 points are *Extra Hard* OTs, which are usually very large trees with many different operations. However, there is a limit to the complexity since it is difficult to come up with questions for OTs with extremely high complexity.

### 4.1.2 Inverted Annotation Procedure

This section presents the inverted annotation procedure, which randomly samples OTs from the context-free grammar and lets humans write the corresponding question, which is answered by the OT. The intuition behind this procedure is that it is easier for humans to understand a query and write the questions instead of inventing questions and writing the corresponding query. This procedure increases the efficiency of the annotation process and ensures more coverage of the database content. In the traditional approach, the generated samples are biased towards the more popular entries in the database (e.g., persons that played in movies).

#### 4.1.2.1 Tree Sampling

First, a set of trees is sampled from the context-free grammar. However, there are various properties, which define how the sampled tree finally looks like. These properties can be manually set or be sampled at random. The option to manually set the properties allows for more fine-grained control over the final dataset. These are the properties:

- *Question Type*: This can be sampled at random or be manually set if a certain type is desired.
- *Result Type*: First, an entity type is randomly sampled from all entity tables in the database. Then a specific set of attributes is sampled from the chosen entity type. Alternatively, the result type can be manually set. Depending on the question type, there are restrictions on which attributes are allowed. For instance, if the question type is a *Sum*, then the result attribute can only be a numeric type.
- *Entity Types*: The entity types are sampled based on the graph structure of the entities and relationships in the database schema. For this, we sample from all the possible join-paths, which contain the table of the result type. This property defines how many *Join* operations are present in the OT. Thus, it has a strong impact on the complexity of the sampled OT. This is also controllable, as we can specify the length of the paths we want to consider. The current implementation only allows to include each entity once, making join-paths that include the same table twice are not yet possible.
- *Constraints*: In the constraints, the *Selection* arguments are sampled. First, the attributes are randomly selected on which the constraints are to be applied. Then we sample an operation and a value at random for each entity type and each attribute. We can limit the number of constraints and the number of maximum constraints for each entity type.

- *Group By*: The *Group By* operations (*AvgBy*, *SumBy*, *CountBy*) are chosen at random or selected manually. For a *Group By* operation, two attributes need to be selected: a group attribute, which defines the attributes to apply the group operation, and an aggregation attribute, which defines on which column to apply the aggregation. For instance, we could group by genre and average over the movie budget, which would state the average budget of each genre.
- *Having*: There is the option to apply a *Selection* on the result of the *GroupBy* operation, which is equivalent to the *Having*<sup>1</sup> operation in SQL. For this, similar to the constraints, the operation and the values are selected at random.
- *Tree structure*: The tree structure is sampled as follows. First, the *Join* operations are applied to the sampled entity types. Second, the set operations (*Union*, *Intersect*, *Diff*) are inserted. Third, the *Selection* operations are inserted. Next, the aggregation operations are inserted, i.e., *Group By*, *Min*, *Max* operations. Finally, the operations for the question type are sampled. For instance, if the question type is a list of entities, we use the *Projection* operation, but if it is a cardinality question, we use the *Count* operation.

This procedure may create trees that make no sense. We handle those trees during the annotation phase, which we describe below. Furthermore, we make sure that the trees are executable. For this, we translate the trees into SQL and run them on the database. We also omit trees that return an empty result, as they can lead to confusion during the evaluation, as two different queries that both return an empty result would be counted as being equal.

Figure 20 shows a randomly sampled tree. During Phase 1 of the annotation procedure, an annotator associated the tree with the question: “*What is the average movie vote of different movies having an Oscar nominee with a cast character called Jesse and were nominated for an Oscar in the year 1991 or later?*”. In the second phase of the annotation, the tokens of the questions were associated with the tree’s nodes. The tree is depicted from root to leaves, where the root node is the last operation, and the leaf nodes are the *GetData*-nodes.

**Example.** Here we describe the tree sampling procedure in more detail with the tree in Figure 20 as an example.

- 1 The query type is selected. There are five different types: *List*, *Sum*, *Count*, *Average*, and *Boolean*. In our example, the average was selected. This can be forced manually or randomly sampled.
- 2 The result type is selected, which, in this case, is *movie.vote\_average*. This

---

<sup>1</sup>Having allows to put filter the outputs of “Group By” functions.

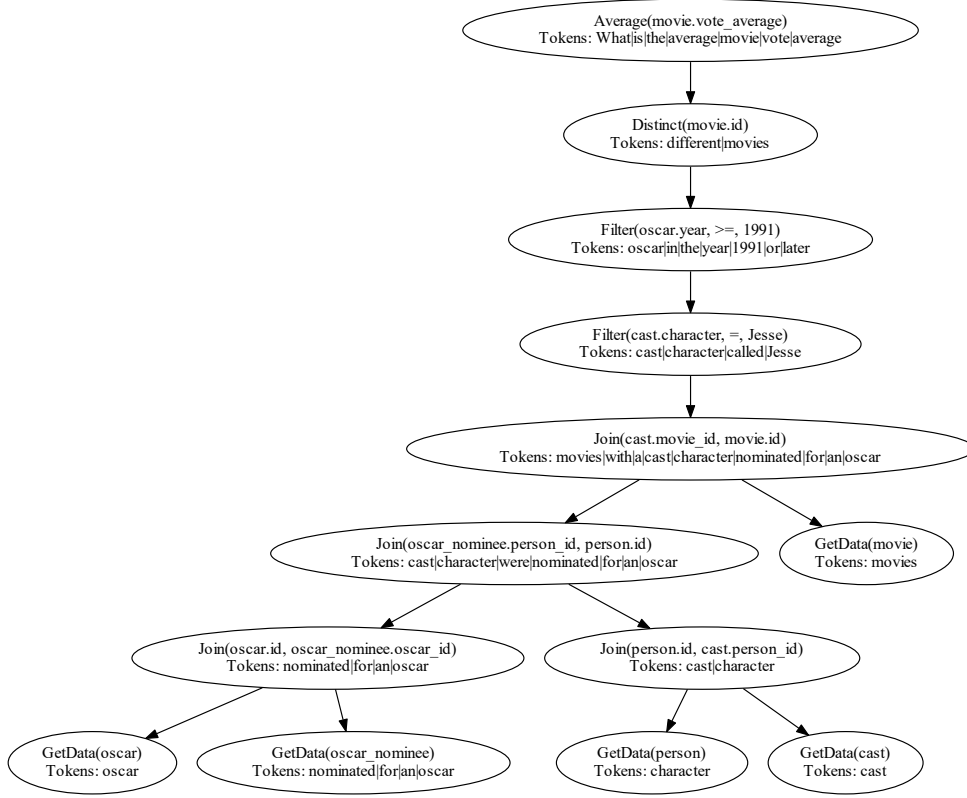


Figure 20: Example of a randomly sampled tree. The nodes denote the node type with their arguments. The *Tokens* are assigned during the second phase of the annotation process. This tree is the answer to the question: *What is the average movie vote of different movies having an Oscar nominee with a cast character called Jesse and were nominated for an Oscar in the year 1991 or later?*. Note that the complexity of this OT is 6 since it contains 4 *Join*-nodes, 2 *Filter*-nodes and an *Aggregation*. Taken from Deriu et al. [2020a]

can also be set manually or be sampled at random. Based on the query type, only certain types of results are allowed. More precisely, for *Average* and *Sum* operations, only numeric result types are allowed.

- 3 The join path is selected. In the first step, a path length is selected, which can be predefined or randomly sampled. Note that a path of length 1 produces trees that work on a single table. In our example, the path length is set to five. Then, in a second step, a random path of the predefined length is selected. That is, a path is created by following the references of the tables in the database (e.g., the cast table refers to movies and persons, thus, enabling



a path from movies to person). In the current example, the query path is:  $\langle \text{movie}, \text{cast}, \text{person}, \text{oscar\_nominee}, \text{oscar} \rangle$ . The path always starts with the result type of table.

- 4 The set operation is selected among *Union*, *Intersection*, or *Difference*. In this example, no set operation was selected. After the operation was selected, a subpath is chosen, on which the set operation is performed. For instance, if we wanted to know the movies where Brad Pitt and George Clooney worked together, then the subpath *movie, cast, person* is selected. Finally, two different filters would be inserted, one for each actor.
- 5 The *Group By* operation is selected. First, the operator is selected among *Sum*, *Average*, or *Count*. Then, the *Group By* attribute and the *Aggregation* attribute are selected. In our example, no *Group By* operation is selected.
- 6 The aggregation operation is selected among the *Min* and *Max* operation. This is relevant for the questions of the type: Which movie has the highest rating. In this example, we have no aggregation operation.
- 7 The filters are selected. For this, we define the number of total filters and the maximal number of filters per table. In this case, we set the number of filters equal to 2 and the maximal number of filters per table to one. Then, the appropriate number of attributes is selected randomly alongside the path. In this case, the tables *oscar* and *cast* were selected. Then, an attribute is selected, followed by a comparison operator and a value randomly sampled from the database. In our example, we have  $\text{oscar.year} \geq 1991$  and  $\text{cast.character} = \text{Jesse}$ .

#### 4.1.2.2 Annotation Procedure

The annotation procedure works in two phases: first, a three-step annotation of the OT with the correct question, and second, the token assignment phase. The annotation tools are shown in figure 23.

**4.1.2.2.1 Phase 1 - Question Annotation** The question annotation phase consists of three steps (see Figure 21 for an Overview).

**Step 1 - Question Annotation.** In the first step, the annotator is shown a randomly sampled OT and is tasked to write the natural language question answered by the OT. Not all randomly sampled OTs can be turned into questions. Thus, the annotators can either skip these OTs or can adjust the Selection operations. For

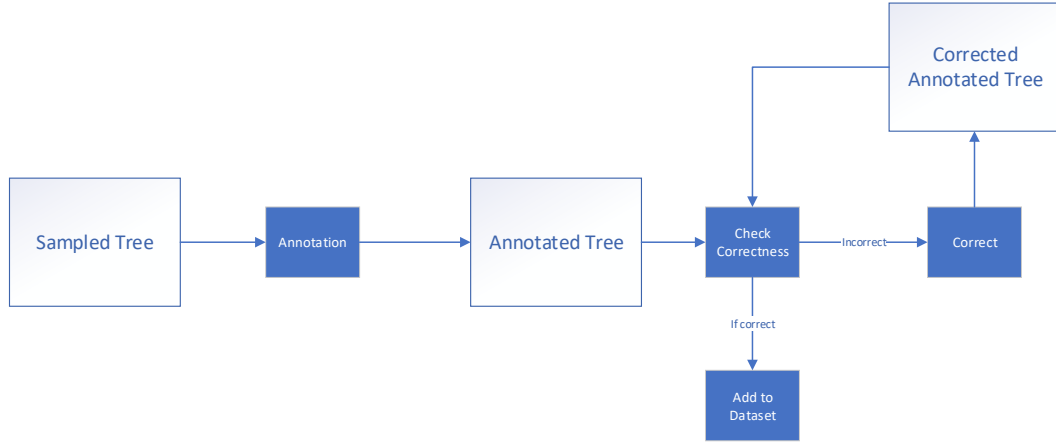


Figure 21: The annotation process of phase 1. First, the OT is sampled from the context-free grammar, then it is shown to the annotators. In step 1, the annotators write the question to the OT. In the second step, the annotators check the question. In the last step, the annotators correct wrongly annotated questions.

instance, if the OT contains two Selections on the same attribute with two equality operations (e.g., `movie.budget = 5000` and `movie.budget = 3500`), then the corresponding question is nonsensical since there cannot be a movie with two different budgets. In this case, it would be sensible to change the comparison operations to *greater than* and *less than* to allow a question about a range of budgets for a movie.

**Step 2 - Question Checking.** In the second step, the annotator is presented with an annotated OT and is tasked to check if the natural language question matches the OT. In case the natural language question is correct, the pair of question and OT is added to the dataset. Otherwise, the annotator marks the question as being wrong, and it is sent to step three. The annotators must also provide a reason for the tree being wrong. There are seven predefined reasons, which cover most of the errors:

1. Grammatical errors and incorrect spelling, where either the sentences are grammatically incorrect, or there are misspellings. The misspellings are especially important for the names of attributes and tables, which need to be written correctly.
2. The M-to-N Problem is best explained with an example. In Figure 22, the OT contains a *Join* between the *keyword* and the *movie* tables via the *has\_keyword* relation table, which form an M-to-N relationship. There is a filter on the `keyword.name` that excludes the keyword "ocean". In this case, an annotator

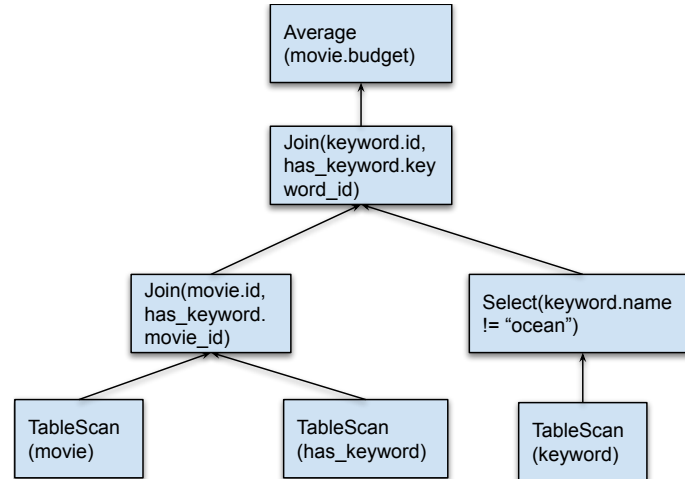


Figure 22: Example of M-to-N problem. The question "What is the average budget of all movies that do **not** contain the keyword ocean?" is not valid since due to the M-to-N relation, movies associated with multiple keywords alongside "ocean" will remain in the result set.

might be inclined to write the question: "What is the average budget of all movies that do not contain the keyword ocean?". However, due to the M-to-N relationship, movies that are associated with multiple keywords will still be in the result set since only the row with the movie and "ocean" is removed. Thus, the aforementioned question is not valid for this OT. Thus, these OTs should have been skipped.

3. Wrong attributes are cases where the natural language sentence includes attributes, which do not occur in the OT or are missing attributes that should occur in the OT.
4. Wrong question types occur when, for instance, the OT contains an *Average* operation, but the question starts with "How many..".
5. Double selection problem describes the aforementioned problem when two *Selection* operations are applied to the same attribute, which exclude each other.
6. Wrong comparison problem occurs when the comparison operation in the *Selection* is incorrectly rendered in the natural language question. Most commonly the *greater than* and *greater than equals* are confused.
7. Min/Max errors occur when the *Min* and *Max* operations are confused or omitted, although they are present in the OT.

**Step 3 - Question Correction.** In the last step, the annotators are presented with an OT and question pair labeled as incorrect in Step 2 alongside the reasons. The task is to correct the question or delete the OT if there is no possibility for correction. In case the OT was corrected, it is sent to revision in Step 2.

**4.1.2.2.2 Phase 2 - Token Assignment** In the second phase, the annotators are presented with a natural language question and OT pair. The task is to assign the tokens of the question to the operations in the OT. Each token can be assigned to multiple operations, and each operation can contain multiple tokens. Since the task is ambiguous, we provided the following guidelines:

- **Table:** If the table denotes an entity type (e.g., *movie*), the tokens that denote this entity type are to be assigned (e.g., "movies"). If the table is a bridge table, which denotes a relationship between entities (e.g., *production\_country*), the tokens that denote this relationship are assigned to the operation. (e.g., "movies", "produced", "in").
- **Joins:** For the *Join* operations, the same guidelines as for the bridge tables are to be followed.
- **Selection:** For the filter constraints (e.g., "person.name= Tom Cruise") the tokens, which represent the constraint, are to be selected (e.g., "by", "Tom", "Cruise").
- **Query type:** For each query type (e.g., *Count*, *Average*, *Sum*, ...), the tokens that correspond or trigger this question type are to be selected (e.g., "How", "many").

### 4.1.3 The OTTA Corpus

We applied our corpus construction procedure to five databases and produced a new corpus with natural language questions and corresponding OTs, called OTTA. This corpus is used for all the experiments presented in this chapter.

In order to compare our results with previous work, we used four databases from the *Spider* corpus (*Chinook*, *College*, *Driving School*, and *Formula 1*), which we extended with a dump from IMDB<sup>2</sup> that we refer to as *Moviedata*. We chose these databases as they contain the most tables and the most entries in the databases, and we deemed them to yield interesting questions. The other databases in *Spider*

---

<sup>2</sup><https://www.imdb.com/>

QUESTIONS

What are the movie genres that are the movie genres that

SubmitSkip Tree

CONSTRAINTS

movie

release\_date

<=

1998-07-29

movie

budget

=

50200000

Execute

NAME

Jan

ANNOTATED DATA

126

SELECT DOMAIN

moviedata

ANNOTATED DATA

0: How many persons were born on November 16

Edit Question

SAMPLED TREE

COMPLEXITY SCORE: 4.5

Done

ExtractValues(genre.name)

Min(movie.vote\_average)

Merge(has\_genre.genre\_id, genre.id)

Merge(movie.id, has\_genre.movie\_id)

GetData(genre)

Filter(movie.budget, !=, 50200000)

GetData(has\_genre)

Filter(movie.release\_date, <=, 1998-07-29)

GetData(movie)

Figure 23: Annotation Tool for step 1.

	MOVIEDATA	CHINOOK	COLLEGE	DRIVING SCHOOL	FORMULA1
#TABLES	18	11	11	6	13
#ATTRIBUTES	64	63	45	39	93
#QUERIES	1148	1067	462	547	568
TIME PER ANNOTATION (SEC)	104	104	77	78	104
AVG. COMPLEXITY	Hard	Medium	Medium	Medium	Medium

Table 15: Statistics of OTTA. Taken from Deriu et al. [2020a].

contain less than five tables and are mostly populated by a couple of entries, which yield empty result sets for most queries. For the annotations, we employed 22 engineers with basic knowledge in SQL-databases<sup>3</sup>. These engineers performed the annotation procedure outlined in section 4.1.2.2.

#### 4.1.3.1 Corpus Statistics

Table 15 summarizes the dataset. The number of tables per database ranges from 6 to 18, and the number of attributes ranges from 45 to 93 columns per database. For *Chinook* and *Moviedata*, our corpus has more than 1000 annotated OTs, while it has around 500 annotated OTs for the other three databases. For *Moviedata*, we also performed the token annotation procedure. For each database, we computed the average *complexity* score. Except for *Moviedata*, which is *Hard*, all other databases have a *Medium* average query complexity. The average time per question annotation ranges from 77 to 104 seconds (average 97.7 seconds) for the first step<sup>3</sup>. The token assignment, on the other hand, took on average 101 seconds per OT.

In the second step, around 66% of the datapoints have been marked for review. In around 25% of the cases, a datapoint was marked as incorrect due to a grammatical error, whereas the other errors were less prominent. In fact, the *min/max* problem and the *double filter* problem occurred only 1% and 3% of the cases. The other predefined reasons occurred in around 10% of the cases. However, the most used reason was the freeform text field, which makes up 56% of the cases. A closer look reveals that it was due to having nonsensical OTs such as OTs, which return the sum over all years in many cases. However, in most cases, the freeform feedback was used to further explain the predefined error list to help the annotators in the

<sup>3</sup>Most of these 22 engineers were students recruited from our university. The others were recruited using <https://www.freelancer.com/>.

<sup>3</sup>Note that we only compare the first step as *Spider* has a different review procedure, which are not comparable to our step 2 and 3.

	#QUESTIONS	#QUERIES	#DB	#TABLE/DB	TABLE COV.	ATTR COV.	MSTTR	AVG. #TOKENS	ANN. TIME
SPIDER	10,181	5,693	200	5.1	0.917 (0.87)	0.621 (0.496)	0.519	12.67	360 sec.
LC-QUAD 2.0	30,000	30,000	1	157,068	0.019	0.187	0.761	10.6	-
OTTA (OURS)	3,792	3,792	5	11.8	0.949	0.544	0.67	13.53	98 sec.

Table 16: Comparison of our corpus OTTA to the Spider and LC-QuaD 2.0 corpora.

We compare the number of different questions (**#QUESTIONS**), the number of different queries (**#QUERIES**), the number of different databases (**#DB**), the number of tables per database (**#TABLE/DB**), the fraction of tables covered by the queries (**TABLE COV.**), the fraction of attributes covered by the queries (**ATTR COV.**), the mean-segmental token-type-ratio (**MSTTR**), the average number of tokens per question (**AVG. #TOKENS**), and the average annotation time for the first phase (**ANN. TIME**). Note that the number of databases in LC-QuaD 2.0 is only 1, since it is an open-domain knowledge base, and the number of tables corresponds to the number of different classes. Numbers in parentheses only consider databases with more than 5 tables. Taken from Deriu et al. [2020a].

third step. For instance, the wrong attribute error was selected with the comment "it's oscar winner instead of oscar nominee".

#### 4.1.3.2 Corpus Comparison

We compare OTTAs characteristics to the *Spider* corpus and the *LC-QuAD 2.0* corpus. We compare the data's coverage, the complexity of the natural language questions, and the complexity of the corresponding SPARQL/SQL queries. *Spider* is a manually generated dataset where the annotators created pairs of natural language questions and the corresponding SQL query. *LC-QuAD 2.0* is an automatically generated dataset, where first SPARQL templates are manually created that are automatically filled with attributes and values. The queries are translated into natural language using a template-based approach. The annotators are then tasked to paraphrase the synthetic questions.

**Coverage.** Table 16 shows the major characteristics of the three corpora. We compare the coverage of the databases in terms of the ratio of tables and attributes that appear in the queries.

The average *attribute coverage* of Spider over all databases equals 62.1%. However, more than half of the databases in Spider contain 5 tables or fewer. Thus, we also report the coverage of attributes only considering the databases with more than 5 tables, where Spider only covers 49.6% of attributes. Corpus OTTA, in contrast,

	#AVG. JOIN	#GROUP BY	#ORDER BY	#NESTED	#HAVING	#SET OP	#AGGREGATIONS	#BOOLEAN
SPIDER	0.537	0.262	0.234	0.148	0.068	0.076	0.519	-
LC-QUAD 2.0	2.05 hops	0	0.041	0	0	0	0.048	0.089
OTTA (OURS)	1.19	0.133	0	0	0.117	0.02	0.4	0.161

Table 17: Comparison of the query complexity based on the ratio of components per query. For the aggregations in LC-QuaD 2.0, we report the number of queries that use a *Count* operation. Taken from Deriu et al. [2020a].

covers 54.4% of all attributes. Furthermore, the divide becomes more apparent when we consider databases with larger amounts of tables. For instance, for the *Formula 1* database, our corpus covers 44.2% of all attributes, in contrast to Spider, where only 22.1% of attributes are covered. LC-QuaD 2.0 covers 1,310 out of 7,005 properties<sup>3</sup> (i.e., attributes in SQL), which corresponds to 18.7%. This is extensive coverage, considering the high amount of properties.

The *table coverage* shows a similar picture: our approach covers 94.9% of all tables, whereas Spider covers 91.7%. This number drops down to 87% when considering only databases with more than 5 tables. Again, this effect is most pronounced for the *Formula 1* database, where we cover 92% of the tables, whereas Spider only covers 69.2%. This shows that our method better scales to larger databases, which is relevant for real-world applications, where databases with a vast number of tables exist. LC-QuaD 2.0 covers around 1.9% of approx. 160k classes, which makes comparison hard, as it is impossible to cover this vast amount of classes with 30k queries.

**Query Complexity.** To compare the complexity of the queries, we examine the number of occurrences of different components in the queries (see Table 17).

We first observe that Spider contains more aggregation operations (in particular *Min*, *Max*, *Count*, *Average*, and *Sum*). This could be easily adapted in our corpus by sampling more trees that contain these aggregations. On the other hand, our corpus stands out in the number of *Joins* per query: on average, OTTA has 1.19 join operations per query compared to Spider, which has 0.537 *Joins* per query. In fact, about 40% of the queries in Spider contain *Joins*, whereas OTTA is composed of 54% of queries, which contain at least one *Join* operation. Furthermore, around 37% of our queries contain two *Joins* in contrast to 9% in Spider. On the other hand, *LC-QuaD 2.0* contains an average of 2 hops (equivalent to two *Joins* in relational

<sup>3</sup>For the number of classes and properties in Wikidata, we consulted: <https://tools.wmflabs.org/sqid>



databases) per query, which lies in the nature of graph database queries that are optimized for handling queries that range over multiple triple patterns. However, *LC-QuaD 2.0* lacks complexity when considering more complex components (e.g., *Group By*, *Set-Operation*, etc.). In addition to the operations in relational algebra, the OTs also support *Boolean* questions (i.e., yes/no questions), which make 16.1% of our corpus compared to 8.9% in LC-QuaD 2.0.

**Question Complexity.** The lexical complexity of the natural language questions is measured in terms of *mean-segmental token-type-ratio (MSTTR)* [Covington and McFall, 2010], which computes the number of different types in relation to all tokens in a corpus. The MSTTR is computed over text segments of equal length to avoid biases due to different lengths within the corpora. First, note that the average length of the questions in all three corpora is approximately the same, between 10.6-13.6 tokens on average. Table 16 shows that our corpus contains a much higher lexical complexity than Spider (0.67 instead of 0.52). Thus, our approach avoids trivial or monotonous questions, which also matches our impression from manual inspection. On the other hand, the lexical complexity is higher in *LC-QuaD 2.0*, which is due to the dataset’s open domain nature.

**Examples.** Table 18 shows examples of questions from OTTA compared to Spider’s questions. The examples show that the quality of the questions is similar. The *easy* questions in both datasets are often only simple filtering questions on one table. *Medium* complexity questions include *Join* operations and filters. *Hard* questions in both datasets include *Join* operations and *Aggregation* operations such as finding the maximum or computing the average. The biggest difference is in the *Extra* complexity. There Spider focuses more on subqueries in the *Where* clause. On the other hand, OTTA focuses more on larger *Join* paths, which are typical for real-world database queries as well as group-by operations and aggregations.

Our analysis shows that the annotation procedure more than triples the velocity of annotation compared to previous methods while maintaining the complexity of the queries and covering a larger part of the underlying databases. The main strength is that it allows generating queries containing relations that span over multiple tables. In fact, 37% of queries in *OTTA* contain two *Join* operations compared to only 9% in *Spider*.

#### 4.1.4 Baseline Model

We have now introduced the OTs and presented the construction of OTTA. To show that the OTs can be used to train an NLIDB system, we present *GrammarNet* in

Hardness	Spider	OTTA
easy	Find the number of albums.	Where were the invoices with the total sum of 1.99 or smaller issued?
	What is the average unit price of all the tracks?	What are the unit prices of tracks composed by Alfred Ellis/James Brown?
	Find all the customer information in state NY.	To which country belongs the 89503 postal code?
medium	Count the number of tracks that are part of the rock genre.	What is the average length of the tracks in the Grunge playlist?
	Please show the employee first names and ids of employees who serve at least 10 customers.	When did we sell tracks larger than 8675345 bytes?
	Find the name of the artist who made the album "Balls to the Wall".	To which postal codes did we sell a track named Headspace?
hard	What is the average duration in milliseconds of tracks that belong to Latin or Pop genre?	How many different playlists with a track that is bigger than 7045314 bytes do exist?
	What are the names of artists who have not released any albums?	What is the album title having the track with the lowest length in milliseconds in the genre name Sci Fi & Fantasy?
	What are the last names of customers without invoice totals exceeding 20?	What are the genres from artists not named Scholars Baroque Ensemble?
extra	What is the name of the media type that is least common across all tracks?	Whats the total unit price sold to customers with the email hholy@gmail.com and Argentina as billing country?
	Count the number of artists who have not released an album.	How many different genres do the tracks have, which were bought by customers who live in France?
	What are the album titles for albums containing both Reggae and Rock genre tracks?	Which customers made at least 35 purchases, excluding titles from the Chico Science & Nacao Zumbi album?

Table 18: Example questions from OTTA and Spider. We grouped the examples by the hardness scores. The examples are for the Chinook domain, which is an online music store database. Taken from Deriu et al. [2020a].

this section. *GrammarNet* is trained on OTTA, and we will use *GrammarNet* for the remainder of this section as the NLIDB system.

*GrammarNet* is an encoder-decoder model, which takes as input a natural language question and outputs a series of production rules that are then synthesized to an OT. Beam-search is used to generate multiple hypotheses during the inference stage, which is relevant for section 4.3. In section 4.1.4.1, we present the model in more mathematical detail, and in section 4.1.4.2, we present the model’s results on the OTTA corpus.

#### 4.1.4.1 Model Description

As a baseline model for OTs from natural language questions, we follow the *Syntactic Neural Model for Code Generation* by Yin and Neubig [2017], referred to as *GrammarNet*. For a more detailed discussion on this architecture, we refer the reader to Yin and Neubig [2017] and section 2.4.1. In our case, it learns to generate the rules defined in Figure 19 for a given question in natural language. Based on the generated list of rules, an OT is synthesized. We extend *GrammarNet* by a token-attention mechanism, which enforces that *GrammarNet* learns the alignments of the tokens to the operations.

We train the model in two phases - a *pre-training phase* and a *supervised phase*. In the pre-training phase, we train a grammar-autoencoder on large amounts of randomly sampled OTs. In the supervised phase, we replace the grammar-encoder with a text encoder and train on the labeled dataset, i.e., the samples with natural language question and corresponding OT.

**Text Encoder.** We use a standard Gated-Recurrent Unit (GRU) [Chung et al., 2014] to encode the natural language question. If  $w_i$  denotes the representation of the  $i$ -th token in the question, then the encoder produces a corresponding hidden state  $h_i^E$ . Let  $H^E \in \mathbb{R}^{N \times h}$  denote the concatenation of all hidden states produced by the GRU for one question, where  $N$  is the number of tokens and  $h$  the size of the hidden state,  $H^E$  will be used in the attention part.

**Grammar Encoder.** The tree encoder, which we use for the pre-training, is based on the same GRU architecture as the decoder. The hidden states for each rule are computed by:

$$h_t = GRU([a_{t-1} : a_{p_t} : n_{f_t}], h_{t-1}) \quad (4.1)$$

In contrast to the encoder, there is no context vector  $c_t$ . Moreover,  $h_{t-1}$  is the last hidden state computed by the GRU. The output of the encoder is a sequence of all states:  $H^R \in \mathbb{R}^{R \times h}$ , where  $R$  denotes the number of rules in the encoded tree.

**Decoder.** The decoder learns to generate a sequence of production rules with which a tree  $y$  is generated for a given encoding  $x$  of the natural language question. The generation process is formalized as:

$$p(y | x) = \prod_{t=1}^T p(a_t | x, a_{<t}, a_{p_t}) \quad (4.2)$$

$a_t$  is the action taken at time  $t$ ,  $a_{<t}$  are the actions taken before time  $t$ ,  $a_{p_t}$  are the

parent actions taken, and  $x$  is the encoded input question.

There are two different types of rules that the model applies during decoding: 1) If the current rule generates a non-terminal symbol, then  $ApplyRule[r]$  is executed, which applies a production rule to the current tree. 2) If the next symbol is a terminal, then  $GenToken[v]$  is applied, which selects the token from a vocabulary. In our case, we have different types of tokens to be generated: table-names, attribute-names, and filter operations. Similar to *GrammarNet*, we implement the decoder using a recurrent neural network, where the internal state is given by:

$$h_t = GRU([a_{t-1} : c_t : a_{p_t} : n_{f_t}], \tilde{h}_{t-1}) \quad (4.3)$$

$n_{f_t}$  is the embedding of the current node type (e.g. average, union, ...),  $c_t$  is a context vector that is computed by applying soft-attention over the input hidden states  $H^E$ , and  $h_{t-1}$  is the hidden vector of the last state. In contrast to Yin and Neubig [2017], we apply attention based on Luong et al. [2015], where  $\tilde{h}_{t-1} = \tanh(W_c[h_{t-1} : c_t])$ .

For the selection of the terms, we have four output matrices  $W_R, W_T, W_A, W_C$ , where  $W_R$  encodes the grammar rules (i.e., for the non-terminal symbols), and  $W_T, W_A, W_C$  encode the table names, attributes, and comparison operations, respectively. Depending on the current node, the probability distribution over all non-terminal rules is computed by:

$$p(a_t|x, a_{<t}, a_{p_t}) = \text{softmax}(W_R * h_t) \quad (4.4)$$

, and assuming that  $ApplyRule[r]$  is the correct rule for the current time-step, the reconstruction loss is computed as,

$$\mathcal{L}_t^{rec} = -\log p(a_t = ApplyRule[r]|x, a_{<t}, a_{p_t}) \quad (4.5)$$

**Token Attention.** A straightforward method to include the explicit token alignment created in the second annotation phase is to force the attention mechanism to learn the alignment. For this, we add an extra loss function, which computes the binary cross-entropy for each attention weight.

More formally, let  $\alpha_t = \text{softmax}(h_{t-1}H^E) \in \mathbb{R}^N$  be the attention weights computed for timestep  $t$ , where  $N$  denotes the number of tokens in the natural language question. Then let  $\alpha_t^{(i)}$  be the attention weight for the  $i$ -th token. We add the following loss function to the reconstruction loss:

$$\mathcal{L}_t^{att} = \sum_{i=0}^N g_i * \log(\alpha_t^{(i)}) + (1 - g_i) * \log(1 - \alpha_t^{(i)}), \quad (4.6)$$

where  $g_i \in [0, 1]$  denotes if the token is assigned to the current node or not. Thus, this loss function forces the attention weights to adapt to the explicit token assignment.

#### 4.1.4.2 Baseline Results

We now report the results of our model. Each experiment is repeated 10 times with different random seeds to create different train-test-splits at random. Similar to Yu et al. [2018b], we use component matching as a correctness measure, which returns true if the generated OT and the gold OT contain the same operations. This allows two OTs with different *Join* orders or different placements of the *Selection* operations to be regarded as equal. We decided against using result-based evaluations since two completely different OTs might result in the same result set, especially when the result is empty <sup>4</sup>. We refer to the component-based evaluation as *CompEq*.

Table 19 shows the precision of the *GrammarNet* on the 5 datasets of OTTA. For the *Moviedata* and the *College* domain, the *CompEq* score lies above 40%, also for the *Chinook* domain scores of almost 40% are achieved. Both the *Formula 1* and the *Driving School* domain achieve scores below 30%. This could be explained by the fact that the *Formula 1* database contains 93 different attributes, and our data only covers 42 of these attributes. Furthermore, each attribute appears only 17.1 times per query on average. In contrast, for the *College* database, the attributes appear in 56 queries on average. Thus, it is harder for the model to learn attributes, which do not often appear in the training set.

In all cases except for the *College* domain, the *Easy* OTs are solved with the highest *CompEq*, and in all cases, the *Extra Hard* OTs achieve very low scores since *Extra Hard* OTs often contain multiple *Joins*, *Aggregations*, and/or *Group By* operators. Note that without the pre-training phase, the scores drop by a large margin. For instance, the scores for *Moviedata* drop below 30% precision. This is because the datasets for the domains are small, and the decoder needs more examples to learn the grammar. Thus, the decoder benefit largely from pre-training.

**Token Assignment Results** We assessed the token assignments’ effect on the systems’ performance. For this, we experimented on the *Moviedata* domain for which we performed the token assignment. We performed an ablation study where we removed the token attention loss function. Unfortunately, we could not measure

---

<sup>4</sup>This might occur when a user asks a question for which there is no result. E.g., "How many movies did Keanu Reeves produce in Russia?" returns "0". The same result is returned by the question: "How many movies, which were released before 1980 did Keanu Reeves play in?". Both questions have the same result and would be treated as equal under the result-based evaluation.

	Easy	Medium	Hard	Extra Hard	Weighted Avg.
MOVIEDATA	0.621	0.535	0.415	0.068	0.424
CHINOOK	0.593	0.275	0.336	0.254	0.395
COLLEGE	0.563	0.574	0.391	0.015	0.449
DRIVING School	0.417	0.176	0.247	0.069	0.294
FORMULA 1	0.392	0.127	0.050	0.000	0.214

Table 19: Precision of queries against our 5 datasets according to query complexity. "Weighted Avg." refers to the mean average precision over all queries irrespective of the query complexity category. Taken from Deriu et al. [2020a].

a significant effect on the final score of the trained parser.

However, we measured a secondary effect. The learned attention is closer to the human-made assignments with the token-assignment loss. More precisely, for each node in the sampled tree, we use the computed attention vector  $\alpha_t \in \mathbb{R}^N$ , and assign each token to the node if  $\alpha_t^{(i)} > 1/N$ , where  $N$  is the number of tokens. To measure how well this automated assignment works, we measure the token-overlap score for each node in the generated OT:  $\frac{|G \cap P|}{|G|}$ , where  $G$  is the set of tokens in the gold node and  $P$  the set of predicted tokens. Averaged over all OTs, the model trained using the token-attention loss has a significantly superior token-overlap score. Concretely, the token-overlap score of the model with attention is at 0.425. In contrast, the model without the token-assignment score achieves a token-overlap score of 0.39.

For example, consider the question "In how many movies did Keanu Reeves play in?". The gold OT contains a *Join* operation between the *movie* table and the *person* table via the *cast* table. Furthermore, assume that the tokens "play" and "in" are assigned to the *Join* nodes in the gold OT. During the decoding stage of *GrammarNet* at step  $t$  when the *Join* node is generated, we measure the attention values in  $\alpha_t$ . At this point  $\alpha_t$  contains an attention value for each of the tokens in the input question. We expect that when generating the *Join* node, the attention values for "play" and "in" are high, whereas the attention values for the other tokens are low. If the value for a token is larger than  $1/N$ , *GrammarNet* assigns the token to the node. Assume that *GrammarNet* only assigned the token "play" to the *Join* node. Then  $G = \{\text{play}, \text{in}\}$ , and  $P = \{\text{play}\}$  yielding,  $\frac{|G \cap P|}{|G|} = \frac{1}{2}$ .

## 4.2 Unreferenced Evaluation based on Back-translation

In the previous section, we introduced operation trees (OT) alongside a more efficient annotation procedure. Furthermore, we introduced *GrammarNet*, which we use as a model for our experiments. The previous discussion was focused on the development stage of the NLIDB model. That is, using a training set to fit the model and a test set to evaluate the model.

In this section, we focus on the deployment stage of the model. During this stage, there is no access to labeled data. Thus, we need a different approach for evaluating the NLIDB system, which does not rely on labeled data. We refer to this type of evaluation as *unreferenced* evaluation.

We propose the following evaluation method, which is based on back-translating a generated OT to natural language and computing the semantic textual similarity score of the user input and the back-translated question (see figure 24 for an overview). Note that we use the definition by Agirre et al. [2016] when we talk about semantic textual similarity. Semantic textual similarity assesses to what degree two chunks of text are similar, usually on a 0-5 scale which ranges from unrelated (0) to semantically equivalent (5).

This section is structured as follows. In section 4.2.1, we introduce the back-translation method, which takes an OT as input and generates a natural language question automatically. Then, in section 4.2.2, we present the unreferenced evaluation method.

### 4.2.1 Back Translation

In this section, we describe the back-translation procedure used to convert OTs to natural language text. This is then used in section 4.2.2 for the unreferenced evaluation method and in 4.3 for the semantic ranking of the hypotheses.

Operation Trees can be translated automatically to a synthetic question by exploiting the context-free grammar and the recursive structure of the OTs. Since statistical methods produce incorrect verbalizations, we opt for a rule-based approach that ensures semantically correct back-translations. This is a crucial requirement for applying the synthetic questions to evaluate and improve the systems' output, which is the topic of sections 4.2.2 and 4.3. The approach is closely related to the *Overnight* approach by Wang et al. [2015], which consists of a domain-agnostic part

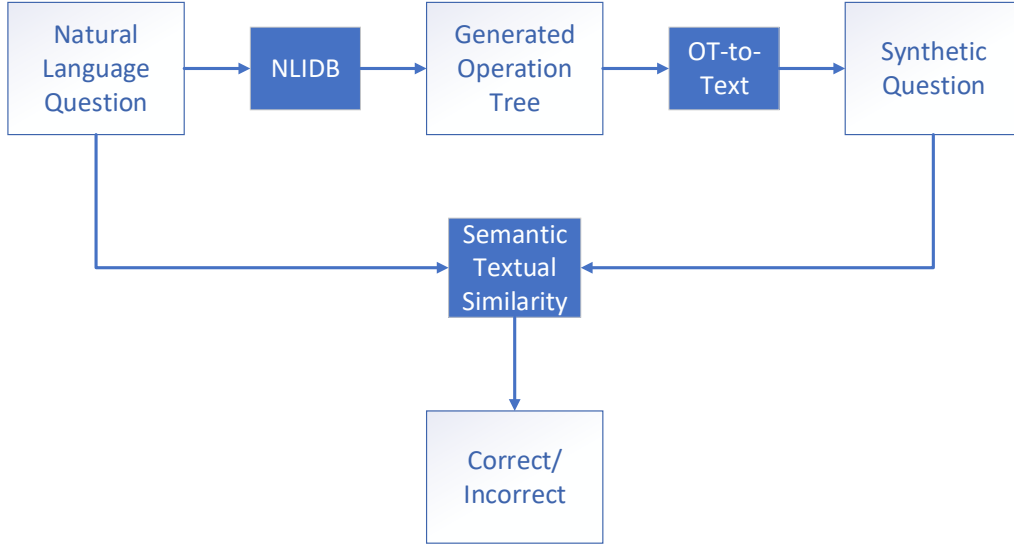


Figure 24: Unreferenced evaluation method. The natural language question is translated to an OT by the NLIDB system. The OT is then back-translated to a natural language question. Then semantic textual similarity is applied to determine if the back-translated question matches the user input.

and domain-specific declarations. We use the OT in Figure 25 as working example<sup>5</sup>. We refer to our approach as OT-to-Text (OT3).

**General Approach.** Generally, the approach works by traversing an OT recursively by applying operation-specific production rules and combining the text fragments from the child operations. The domain-specific data influences the word choices at specific nodes (e.g., "played in" describes the "cast" relation). The overall composition of the fragments depends on the tree structure.

**General Production Rules.** In Table 20, the general production rules are listed. These rules are domain-agnostic and define the overall structure of the generated question. For instance, the rule for the *Sum* operation expands the attribute name, over which the sum is computed, and the table, which is the result of applying various *Join* and *Selection* operations.

**Domain-Specific Rules.** The domain-specific rules need to be provided by a domain expert. The data consists of the names of all attributes and tables. The type of attribute influences how selection operations are expressed. For instance, the *greater than* operation is expressed as "later than" for attributes of type date and expressed as "bigger than" for an attribute that stored measurements (see Table

<sup>5</sup>Note that the back-translations were not implemented for *GroupBy*- and *Set*-Operations.



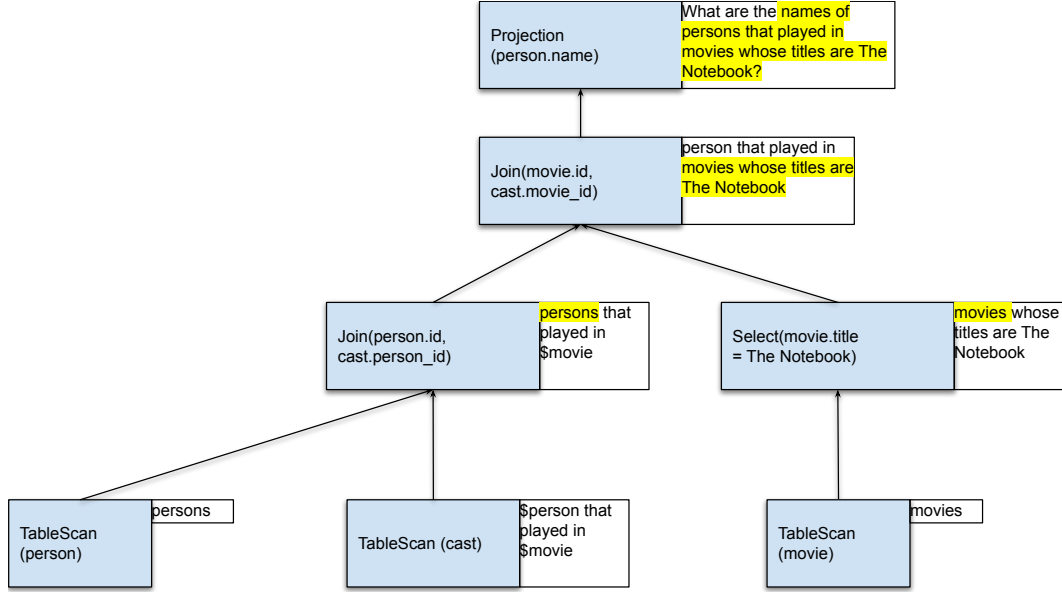


Figure 25: Example of the construction of the natural language question from an OT. We denote placeholders of subtrees  $T$  with  $\$T$ , which will be filled in by sister subtrees. For instance,  $\$movie$  is replaced by the phrase "movies whose titles are The Notebook". The replacements of placeholders are denoted in yellow. Note that the templates do not distinguish singular and plural cases. Thus, even if there is only one movie called *The Notebook*, the template uses the plural (i.e., "movies") as it does not access the database.

root nodes	
SUM	$[[sum(T, A)]] = \text{What is the total } [[A]] \text{ of } [[T]]?$
AVERAGE	$[[average(T, A)]] = \text{What is the average } [[A]] \text{ of } [[T]]?$
COUNT	$[[count(R)]] = \text{How many } [[R]] \text{ are there?}$
ISEMPTY	$[[isEmpty(R)]] = \text{Are there any } [[R]]?$
DONE	$[[done(R)]] = \text{What are the } [[R]]?$
projection	
PROJECTION	$[[projection(T, A)]] = [[A]] \text{ of } [[T]]$
aggregations	
MIN	$[[min(T, A)]] = [[T]] \text{ with minimum } [[A]]$
MAX	$[[max(T, A)]] = [[T]] \text{ with maximum } [[A]]$
distinct	
DISTINCT	$[[distinct(T)]] = \text{distinct } [[T]]$

Table 20: Overview of domain independent production rules, where  $[[N]]$  indicates the expansion of some node  $N$ .

<b>Generic</b>		
[[ <i>T</i> ]]	whose [[ <i>A</i> ]]	is
		is not
		[[ <i>LIT</i> ]]
<hr/>		
<b>Numeric</b>		
		of
		other than
[[ <i>T</i> ]]	with a [[ <i>A</i> ]]	of more than
		of at least
		of less than
		of at most
		[[ <i>LIT</i> ]]

Table 21: Overview of production rules for selections for numerical and generic attributes. Similar rules can be defined for other types of attributes. The expression  $[[A]]$  expands to the name of the attribute (e.g. `movie.title`  $\rightarrow$  "title") and  $[[LIT]]$  expands to the literal (e.g. "The Notebook").

21 for an example). Thus, the domain expert needs to provide the names for the attributes and how the comparisons are expressed for different types of attributes. For the tables, there are *entity* and *relation* tables. For entity tables, the domain expert needs to provide how these are expressed in natural language.

The main strength of our approach is its ability to express relations that range over multiple tables. For each relation, the domain expert needs to define how they are expressed in natural language. Here, both directions must be covered. For instance, the relation "cast" can be expressed as "\$person that played in \$movie" or "\$movie where \$person played in". Both options must be provided since the OTs allow for both directions.

**Discussion.** As we mentioned, our approach is closely related to the Overnight approach. However, Overnight mainly works on simple triples of the form (*subject*, *predicate*, *object*), whereas OT3 works on complex queries over a relational database. In particular, it allows queries with a large amount of *Join* operations.

OT3 also lies in contrast to template-based approaches [Zhong et al., 2018; Yu et al., 2018a], which manually predefine templates for SQL queries and their respective natural language questions and replace the literals. OT3 can generate a natural language question for every OT, generated from the context-free grammar.

Analogous to the Overnight approach, a database expert needs to provide how tables, attributes, and relations are rendered in natural language, a one-time effort.

Thus, our approach is easily transferable to other domains<sup>6</sup>. Furthermore, we do not rely on labeled data to train OT3. In fact, a neural network-based approach fails, as it requires large amounts of data, which is not realistic in this setting<sup>7</sup>. In contrast to statistical approaches, ours also ensures that the synthetic questions are semantically correct in most cases, which is crucial for our use case.

## 4.2.2 Evaluation Method

In this section, we present the evaluation method, which is based on back-translation and semantic textual similarity.

For this, let  $Q_h$  be the user input,  $OT_p$  the generated OT by the NLIDB system,  $Q_s$  the back-translated question generated by OT3 (see Section 4.2.1), and  $OT_g$  the ground truth. We first need to show that  $Q_s$  correctly expresses the semantics of  $OT_p$ . Then, we show that  $OT_p \underset{c}{=} OT_g \leftrightarrow Q_h \underset{s}{=} Q_s$ , where " $\underset{s}{=}$ " stands for semantic equivalence, and " $\underset{c}{=}$ " stands for query equality. That is, the generated OT is correct with respect to the gold OT if and only if  $Q_h$  and  $Q_s$  are semantically equivalent.

### 4.2.2.1 Experimental Setup.

In order to compute the semantic textual similarity between two questions, we apply NUBIA [Kane et al., 2020], a pre-trained model that scores a pair of sentences based on their interchangeability. We use NUBIA<sup>7</sup> out-of-the-box without any fine-tuning. It extracts features from RoBERTa and GPT-2 [?] and fine-tunes a fully connected network to output a score between 0 and 1, indicating how interchangeable two input sentences are. We will use NUBIA to automatically score the similarity between a natural question and a back-translated question throughout this work. For the NLIDB system to generate OT's, we used *GrammarNet* as described in Section 4.1.4. We trained it for each domain on 60% of the data, and used 20% as a development set to determine the NUBIA threshold for deciding semantic equivalence, and used 20% as the test set. For each sample  $Q_h$  in the test set, we used *GrammarNet* to generate  $OT_p$  and OT3 to create  $Q_s$  out of  $OT_p$ . We then applied NUBIA to compute the semantic similarity between  $Q_h$  and  $Q_s$ : If the score was above the threshold, the pair was labeled semantically equivalent. The threshold is computed on the 20%

---

<sup>6</sup>For our domains with databases of around 10-20 tables, this requires 2 hours of effort per domain. A database with over 1000 tables will require a greater effort.

<sup>7</sup>We experimented with a neural approach, however, the generated questions were of low quality, which is mainly due to the lack of large training datasets.

<sup>7</sup><https://github.com/wl-research/nubia>

User Input	Back-translation	Crowdworker	NUBIA	CompEq	Complexity
What is the popularity of movies released on the date 1995-07-07?	What are the popularities of movies who were released on 1995-07-07?	True	True	True	Easy
Which movies have been produced in Lithuania?	What are the names of movies whose status is Lithuania?	False	False	False	Medium
Which companies produced their films in Serbia and Montenegro?	What are the names of companies which produced movies which were produced in countries whose name is Serbia and Montenegro?	False	True	True	Hard
Do countries exist where movies were produced that have an oscar nominee who was born before May 27th 1922?	Are there any names of countries in which movies for which people who were born before 1922-05-27 were nominated for oscars were produced?	True	True	True	Hard

Table 22: Examples of human written questions (user input), and generated questions (back-translations), which represent the structured query produced by the NLIDB system. Alongside are the judgements of crowdworker and the NUBIA system regarding semantic similarity, the ground truth correctness (CompEq) for the semantic parse, and the query complexity.

development set such that it maximises the overlap with *CompEq*. For comparison, we also report how well the BLEU score [Papineni et al., 2002] performs to determine semantic equivalence as well as a human evaluation, where humans state if  $Q_h$  and  $Q_s$  are semantically equivalent.

The human evaluation is performed by recruiting crowdworkers on Mechanical Turk. The crowdworkers were presented only with  $Q_h$  and  $Q_s$ , for which they had to decide if they were semantically equivalent or not. We let three crowdworkers annotate each pair and determine semantic equivalence based on either *majority voting* or *strict voting* (i.e., if one worker states that  $Q_h$  and  $Q_s$  are not equivalent, then the pair is labeled as not equivalent). The correctness of the generated OT with respect to the gold label is measured using the component-based method (*CompEq*).

Table 22 shows examples of user inputs and back-translations. The back-translations are generated by applying OT3 to the output of *GrammarNet* for the given user input. The table also shows the crowdworker annotation, that is, if crowdworkers think the two utterances are semantically equivalent. It shows the NUBIA output, that is if the NUBIA score is above the determined threshold. *CompEq* shows if the generated OT and the gold OT are equivalent. Ideally, the crowdworkers and the NUBIA entries correspond to the *CompEq* entry. That is, if the generated OT is correct (according to *CompEq*), the semantic equivalence should also be given (according to the crowdworkers and NUBIA) and vice-versa.

Domain	System	Precision	Recall	F1	Acc
Chinook	Nubia	0.712	0.846	0.773	0.734
	BLEU	0.616	0.906	0.734	0.647
	Human Majority Vote	0.854	0.701	0.770	0.775
	Human Strict Agreement	0.697	0.906	0.788	0.739
Moviedata	Nubia	0.715	0.807	0.759	0.716
	BLEU	0.573	0.972	0.734	0.584
	Human Majority Vote	0.955	0.578	0.720	0.751
	Human Strict Agreement	0.774	0.817	0.795	0.766

Table 23: Performance of detecting parsing errors of different predictors for both the *Moviedata* and *Chinook* domains.

#### 4.2.2.2 Results

Table 23 shows the results for both domains. Here, we measure how often the output of the semantic similarity system NUBIA agrees with *CompEq*, which measures the equivalence to the generated OT and the gold standard OT. NUBIA achieves an Accuracy score of 0.734 and 0.716 for the respective domains with respect to *CompEq*, which is comparable with the human performance on the task (rows Majority Vote and Strict Agreement in Table 23). This means that in over 71% of all cases, the semantic equivalence metric agrees with the *CompEq* metric, which shows that semantic equivalence is an approximation for the direct comparison to the gold label (i.e., *CompEq*). For comparison, the BLEU score only agrees in 64.7% and 58.4% of cases with the *CompEq* metric. This indicates that the word overlap is not sufficient to approximate *CompEq*.

A more in-depth analysis reveals the strength and weaknesses of this approach. We also report in table 23 the precision and recall scores of the "incorrect" label of the semantic similarity methods. Thus, recall denotes the fraction of incorrect parses, which are found by textual semantic similarity methods. For instance, NUBIA achieves a recall of 0.846 in the Chinook domain, which means that 84.6% of all incorrect parses are labeled incorrect. On the other hand, precision shows the fraction of cases where a sample is correctly labeled as incorrect. For instance, of all samples labeled as incorrect by NUBIA in the Chinook domain, 71.2% are actually incorrect, which means that 29.8% of the samples labeled as incorrect are actually correct. Both NUBIA and BLEU tend to have better recall than precision, i.e., they tend to be conservative in the sense that they tend to label a sample as incorrect.

However, this is due to the selection of the threshold. A more liberal threshold would lead to higher precision and lower recall. The human raters obtain slightly better results than the NUBIA-based evaluation. However, depending on how the three ratings are aggregated, a different picture emerges. When using *majority voting*, the precision is significantly higher than when using *strict voting*. This is expected as the strict agreement requires only one judge to label the sample as semantically nonequivalent, resulting in more samples being rated as such.

**Precision/Recall Space.** To better compare human performance with the automated methods, we show the Precision-Recall Space in Figure 26. It shows the Precision-Recall (PR) curve for both NUBIA and BLEU and the selected operating point, based on a decision threshold tuned on the development set. For both domains, the performance of NUBIA is superior to BLEU. The figure also shows the performance of the single annotators who annotated more than 20 annotated samples. While most annotators outperform NUBIA, there are some that lie below the PR curve. We see that most single annotators achieve higher precision than recall. That means that they do not label a clearly wrong parse as correct. However, the lower recall indicates that humans mistakenly label correct parses as wrong. In both domains, the best annotators distinguish themselves with a higher recall than their peers.

**Sanity Check.** To show that  $Q_s$  correctly renders the semantics of an  $OT$ , we first perform a sanity check, where we back-translated the gold-standard tree  $OT_g$  for a given question  $Q_h$ . Thus, we need to show that  $Q_h$  and  $Q_s$  are semantically equivalent. As negative examples, we also mix in randomly sampled human questions  $Q_n$  for a given  $Q_h$ . Thus  $Q_h$  and  $Q_n$  should never be semantically equivalent. We let humans annotate this data. In this setting, humans agree in 94% of cases with the parsing ground-truth. This shows that the synthetic questions are understandable and generally maintain the semantics of the underlying  $OT$ . As an additional measure, we computed the BLEU [Papineni et al., 2002] score between  $Q_h$  and  $Q_s$ . It achieved a score of 0.59 and 0.56 for *Chinook* and *Moviedata* respectively. This shows that on the lexical level, the synthetic questions are similar to their human counterpart.

We showed that computing the textual semantic similarity score between  $Q_h$  and  $Q_s$  either manually or automatically can be used for evaluating the underlying  $OT$ 's correctness.

## 4.3 Back-translation and Semantic Textual Similarity for Re-ranking

In the previous section, we discussed the application of back-translation in combination with semantic textual similarity for evaluating an NLIDB system without labeled data. The results indicate that the back-translated questions are of high quality.

In this section, we go one step further. We apply back-translation coupled with semantic textual similarity to improve the NLIDB system. We propose two approaches. First, we apply back-translation for data augmentation, and second, we apply back-translation coupled with semantic textual similarity to re-rank the set of hypotheses of the beam search.

### 4.3.1 Data Augmentation

The nature of OTs allows for a straightforward two-step data augmentation method. First, an OT is randomly sampled from the context-free grammar (see section 4.1.2.1). Second, back-translation is applied to translate the sampled OT to a natural language question. This process allows the creation of a dataset of arbitrary size.

Since the generated questions do not display the same variety as human-written text, we use the automatically generated dataset for a pre-training step. More precisely, we automatically generate a large training set and use it to train *GrammarNet*. In a second step, we fine-tune *GrammarNet* using *OTTA*.

### 4.3.2 Ranking Hypotheses based on Semantic Textual Similarity

*GrammarNet* works by applying a beam-search algorithm for inference generating a set of hypotheses. These hypotheses are ranked with regards to the confidence scores that are emitted by the decoder. This ranking is not optimal. In fact, as we will see later in more detail, it often ranks the correct OT lower than an incorrect OT. Thus, we propose to apply back-translation coupled with semantic textual similarity to re-rank the set of hypotheses.

More precisely, given a human input question  $Q_H$ , the beam search decoder generates a set of  $k$  hypotheses:  $H = \{T_{H_1}, \dots, T_{H_k}\}$ . The baseline (confidence) ranking for a

hypothesis is computed as  $c_i = \frac{1}{K} \sum_k -\log(p(y_k|y_{\leq k}, x))$ .

To re-rank those hypotheses, we apply back-translation to generate the synthetic question for each OT in the set of hypotheses. Then we apply NUBIA to compute the semantic similarity score between the synthetically generated question and the original question. More precisely, each hypothesis in  $H$  is translated to a synthetic question  $Q_{gen_1}, \dots, Q_{gen_k}$  using back-translation. For each synthetic question  $Q_{gen_i}$ , the semantic similarity to  $Q_H$  is computed using NUBIA:  $s_i = NUBIA(Q_H, Q_{gen_i})$ . Finally, the set of hypotheses is ranked with respect to the similarity scores. We refer to this ranking as *Semantic only* (figure 4.3 depicts the process).

Our experiments have shown that *Semantic only* and *Baseline* ranking disagree in 16% – 21% of cases. That is, in these cases, one method ranks first the correct hypothesis while the other ranks first the wrong hypothesis. Thus, we propose a combined approach, which is based on the multiplication of the *Semantic only* and the *Baseline* scores  $m_i = s_i * c_i$ , which we refer to as *Semantic full*. Since  $0 \leq s_i \leq 1$ , the *Semantic full* score can be interpreted as a weighted confidence score.

Based on these observations, we finally compared four different re-ranking schemes: (1) *Baseline* ranking, (2) *Semantic only* ranking, (3) *Semantic full* ranking, and (4) *Oracle* ranking, which returns the correct hypothesis if it is present in the set of candidates. The Oracle ranking serves as an upper bound to the potential of hypothesis re-ranking.

### 4.3.3 Experimental Setup

We apply the pre-train and re-rank procedure to the *GrammarNet* described in Section 4.1.4.

In *GrammarNet*, the model consists of a text-encoder and a decoder, which produces a series of grammar-production rules. During inference, beam search is applied to generate  $k = 15$  hypotheses<sup>8</sup>.

**P1 - Pre-training:** We train the text-encoder and tree-decoder architecture on the automatically generated data. For this, we use the augmented dataset, which we created according to the process described in section 4.3.1.

**P2 - Fine Tuning:** We fine-tune the model from P1 using the human-annotated data. For this, we train the model using early stopping and use an 80/20 train-test split.

---

<sup>8</sup>We also experimented with larger and smaller beam-sizes. The results suggest that increasing the beam size over  $k = 15$  does not increase the performance significantly. In fact, there is no difference in score between  $k = 15$  and  $k = 20$



	Chinook		Moviedata	
	Normal	With Pretraining	Normal	With Pretraining
Easy	0.42	0.53	0.51	0.55
Medium	0.38	0.44	0.49	0.52
Hard	0.27	0.33	0.28	0.32
All	0.35	0.43	0.43	0.47

Table 24: Accuracy results of pretraining for both the *Chinook* and *Moviedata* domain. We differentiate with regards to the complexity scores of the OTs.

We repeat the process 10 times in order to guarantee reliable results. For each repetition, we produce a different train-test split, which is stratified with regard to the complexity of the queries.

**Data.** We apply this process to the *Moviedata* and *Chinook* domain of the *OTTA* corpus. For pre-training, we generated 5000 new samples for *Moviedata* and 3000 samples for *Chinook*. For the fine-tuning, we used 992 datapoints for *Chinook* and 892 datapoints for *Moviedata* to train. For testing, we used 248 and 223 datapoints for *Chinook* and *Moviedata* respectively.

Tables 24 and 25 display the accuracy results for pre-training and re-ranking for both the *Chinook* and the *Moviedata* domain. Accuracy is defined as the percentage of times where  $OT_g = OT_p$  that is the number of times that the generated OT and the gold OT are equivalent according to *CompEq*.

**Pre-training Discussion.** The results are shown in table 24. We compare the model without pre-training to the model that was pre-trained with the synthetic data. As expected, for both domains, the pre-training is beneficial. In fact, the score increases by 8% and 4% for *Chinook* and *Moviedata*, respectively. A closer look reveals that pre-training especially benefits the easy questions in the *Chinook* domain, where the increase in score lies between 8 – 10%. In the *Moviedata* domain, each complexity class benefits similarly from pre-training, but the largest increase here is for hard questions.

**Re-ranking Discussion.** Table 25 shows the results for the re-ranking. First, we note that there is a large discrepancy between the *Oracle* and the other strategies. This indicates that there is still a large margin for improvement by finding better re-ranking schemes. In fact, between the *Baseline* and the *Oracle* approach lie between 18% – 20%. However, using just the Semantic re-ranking, the score increases by

	Chinook			
	Easy	Med	Hard	Tot
Baseline	0.53	0.44	0.33	0.43
Semantic only	0.50	0.46	0.38	0.45
Semantic full	0.59	0.52	0.40	0.51
Oracle	0.70	0.67	0.49	0.61

	Moviedata			
	Easy	Med	Hard	Tot
Baseline	0.55	0.52	0.32	0.47
Semantic only	0.54	0.50	0.35	0.47
Mixed	0.61	0.58	0.39	0.53
Oracle	0.73	0.70	0.51	0.65

Table 25: Accuracy results of re-ranking for both the *Chinook* and *Moviedata* domain. We differentiate with regards to the complexity scores of the OTs.

2% – 4% in the *Chinook* domain. For *Moviedata* the increase lies at 3% for the non-pre-trained model. Interestingly enough, this domain’s pre-trained model does not benefit from the *Semantic only* re-ranking. A more detailed analysis revealed that the *Semantic only* and *Baseline* approaches are not entirely correlated. That is, in around 20% of cases, one of the two approaches returns a wrong result, whereas the other returns the correct result. This is the reason why the *Semantic full* approach is beneficial in all cases. For the *Chinook* domain, we see an increase of 8 – 10% and in the *Moviedata* the improvement lies at 6% – 9%.

#### 4.3.4 Results

**Learning Curve Analysis.** We also investigated the influence of the amount of available fine-tuning data. In Figure 28, the learning curves for both domains are shown. The results show that in the low-data regime, the pre-training shows a large effect. Without the fine-tuning, both domains achieve between 4% and 7% accuracy. With 20% of the training data, pre-training the model is more beneficial than re-ranking. In fact, for *Moviedata* it increases the score by 14 – 16 accuracy points. The same pattern is apparent for *Chinook*. However, when using more than 40% of the data, the re-ranking results have a stronger effect. In fact, the *Semantic full* ranking without pre-training achieves a higher score than the *Baseline* ranking with pre-training. The effects of pre-training vanish with higher amounts of fine-tuning

$Q_h$ :	Whats the average track size of tracks purchased from 120 S Orange Ave?				
<b>i</b>	$Q_p$	$c_i$	$s_i$	$m_i$	CompEq
1	What is the average size of all tracks on invoice lines which are part of invoices?	-0.4018	0.49	0.29	False
2	What is the average size of all tracks on invoice lines which are part of invoices whose billing street is 120 S Orange Ave?	-0.4022	0.61	0.36	True
9	What is the average size of all tracks on Albums on invoice lines which are part of invoices whose billing street is 120 S Orange Ave?	-0.458	0.3	0.16	False
$Q_h$ :	Which companies from Mexico produced their films in Mexico ?				
<b>i</b>	$Q_p$	$c_i$	$s_i$	$m_i$	CompEq
1	What are the names of companies which produced movies whose status is Mexico?	-0.316	0.676	0.462	False
3	What are the names of companies which produced movies which were produced in countries whose name is Mexico?	-0.339	0.751	0.496	True
5	What are the names of companies which produced movies whose name is Mexico?	-0.409	0.741	0.437	False

Table 26: Cherry Picked example of reranking. We show two original queries ( $Q_h$ ) and three backtranslated examples of the 15-best list for each.  $c_i$  for decoder score,  $s_i$  for similarity score,  $m_i$  the combination, CompEq for whether the parse is correct. In both cases the combination scores highest the correct parse.

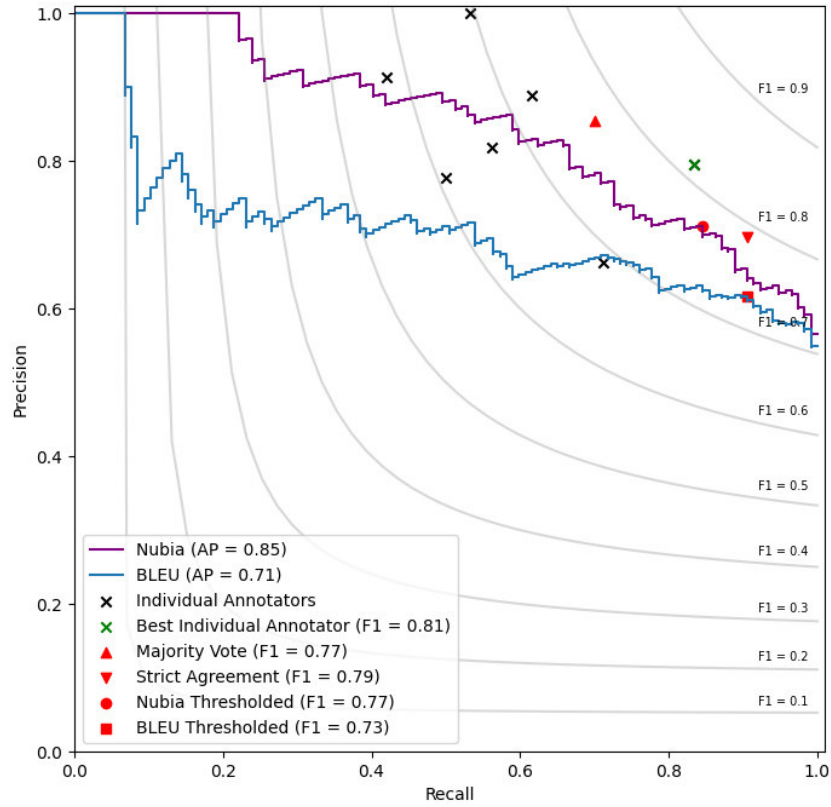
data. For the *Moviedata* domain, the effects of pre-training almost vanish when using more than 60% of training data.

**Qualitative Analysis.** In Table 26, we show examples of the different rankings. We show two representative examples of sets of hypotheses. In the first example, we note that the hypothesis with the best confidence score is incorrect. According to the confidence score, the second-best hypothesis is correct and has a very similar score to the first placed (-0.4018 *vs.* -0.4022). The hypothesis placed 9<sup>th</sup> adds an unnecessary relation. However, the confidence score is still close to the first placed hypothesis. On the other hand, the semantic score is more accurate. The correct hypothesis is placed 1<sup>st</sup> with a large margin (0.61 *vs.* 0.49) and an even larger difference to the 9<sup>th</sup> place. The second example shows a similar pattern: the first hypothesis is obviously wrong. The second hypothesis, which is correct, gets a much lower confidence score. The *Semantic only* score ranks the set of hypotheses correctly. However, *Semantic only* re-ranking alone is not enough; the 5<sup>th</sup> ranked example has a high semantic similarity score while being incorrect. In this case the *Semantic full* approach helps differentiating: While  $s_3$  and  $s_5$  are very close,  $m_3$  and  $m_5$  have a bigger margin.

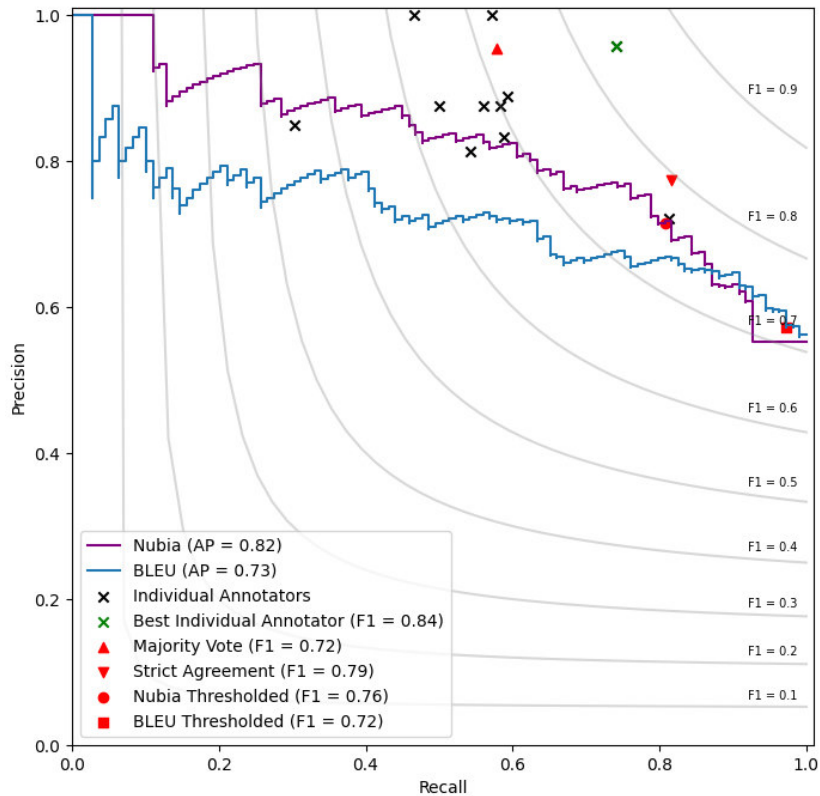
## 4.4 Summary

This chapter presented three main contributions. First, a more efficient annotation procedure, second, an unreferenced method for evaluating an NLIDB system after deployment, and third, an application of the metric for improving the system.

The results show that the inverted data annotation procedure is more efficient and yields the same complexity of questions as the traditional approach. This is because the annotators only need to write the question for a given OT. Our new approach allows designing the dataset by stating the queries' properties, which should be annotated. The representation of the queries as OTs allows for a straightforward implementation of a back-translation algorithm. To adapt the back-translation algorithm to a new domain, a domain expert needs only to provide the domain-specific information. We apply the back-translation and semantic textual similarity to create an unreferenced metric, which we showed to be a proxy for the gold-standard-based evaluation. This is especially interesting for scenarios with no gold standard available, e.g., when monitoring a deployed system's performance. In this case, the user is shown the back-translation of the generated OT, which the user can compare to the input question. Lastly, our experiments show that using back-translation and semantic textual similarity can improve the system by creating synthetic data and re-ranking the beam search hypothesis using semantic similarity scores.



(a) Chinook.



(b) Moviedata.

Figure 26: Precision Recall Space for the *Moviedata* and *Chinook* domain. It shows the human performance for each annotator who annotated at least 20 samples. We also show the scores of the Majority Vote, the Strict Vote, as well as NUBIA and BLEU with their selected thresholds.

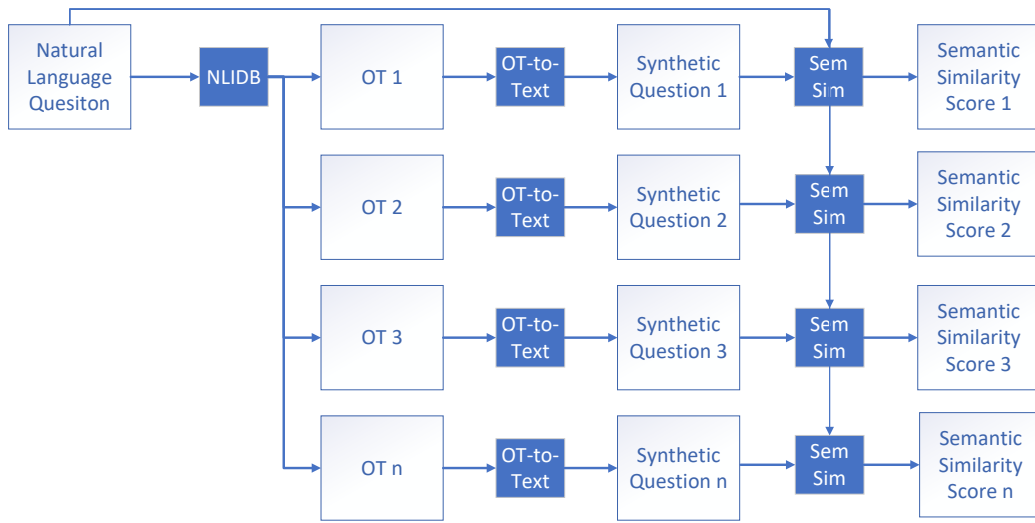


Figure 27: Hypotheses ranking in chapter 4.3. For a natural language question, the NLIDB system generates a set of  $n$  hypotheses using beam-search. Each of these hypotheses is back-translated to a natural language question, which we call the synthetic question. Each synthetic question is compared to the original input using a textual semantic similarity tool. The hypotheses are ranked according to the semantic similarity score.

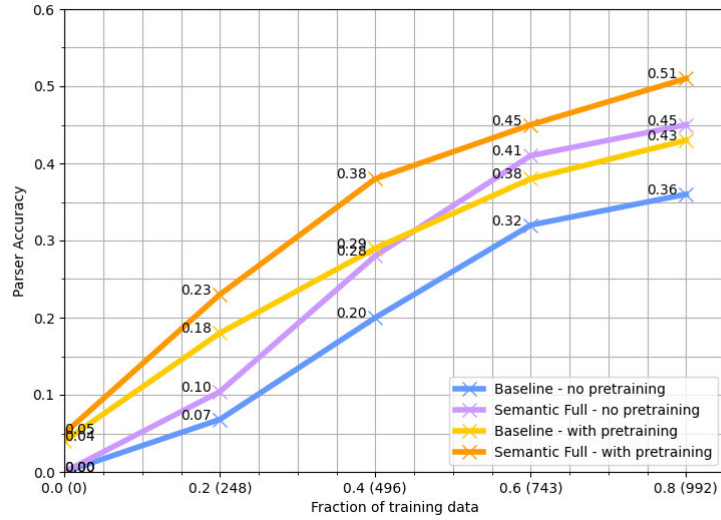
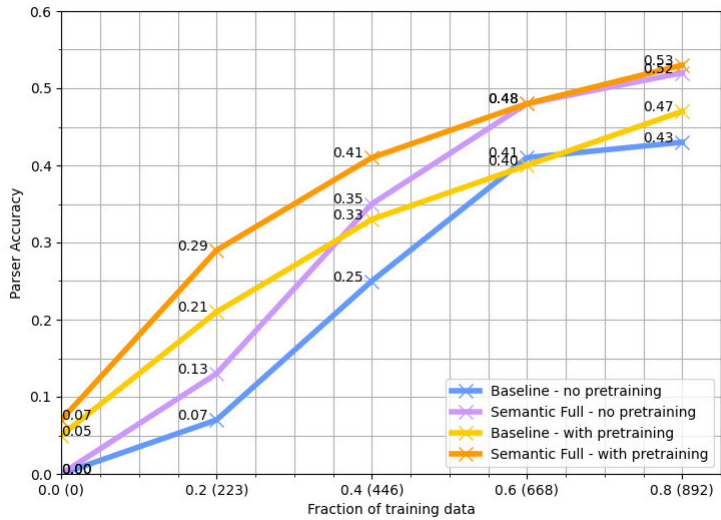
(a) *Chinook*.(b) *Moviedata*.

Figure 28: Learning Curves for *Chinook* and *Moviedata*. We show the scores for both the non-pre-trained models and the pre-trained models, with the Baseline reranking and the *Semantic full* ranking. In parentheses, we include the absolute number of training samples.

## 5 Conclusion and Outlook

In this thesis, we studied the evaluation of dialogue systems. We focused on two types of dialogue systems: conversational dialogue systems (i.e., chatbots) and natural language interfaces to databases. Both systems pose different challenges to the evaluation. The conversational dialogues systems are hard to evaluate due to the difficulty of measuring criteria of quality reliably, making current evaluation procedures unreliable. On the other hand, NLIDBs are straightforward to evaluate, thus, we investigated leveraging the evaluation to improve the system.

For conversational dialogue we introduced a novel paradigm for evaluation, namely bot-to-bot talk. This allows automating the dialogue generation process, which drastically reduces the effort and enables analysing the multi-turn behavior of dialogue systems. Based on this, we present two evaluation procedures. First, Spot The Bot that uses these automatically generated dialogues as a basis for evaluation. The evaluation is performed regarding the human-likeness of the system. Spot The Bot is shown to be efficient and robust. The second evaluation method is AutoJudge, a trained metric that is based on annotating self-talk dialogues on a turn basis. These annotations are used to train a regression model, i.e. AutoJudge, which is shown to correlate well to human judgements and can be used as answer selection module. However, AutoJudge also showcases the fundamental problems of trained metrics. Namely, it is not robust and can easily be fooled.

For NLIDB we introduced a novel method to efficiently gather data. The method is based on inverting the annotation procedure, i.e., by first sampling the query from a context-free grammar and then letting humans write the question. Furthermore, we introduce a novel representation for the query, based on abstract syntax trees. We apply this method to create the OTTA corpus, consisting of 3500 queries spanning 5 different domains. We introduced a novel evaluation procedure to evaluate a NLIDB system after deployment. For this, we leverage the back-translation of a generated query and use textual semantic similarity to measure the correctness of the query. We showed that it can be used to evaluate a NLIDB system. Based on this procedure, we proposed an improvement scheme based on hypothesis re-ranking, showing a 10 to 16 point increase in score.



In the following subsections, we give an overview of potential future work. First, for the conversational dialogue systems, and then for the NLIDB systems.

## 5.1 Conversational Dialogue Systems

The evaluation of conversational dialogue systems is still an open problem. This thesis provides one main contribution to solving this problem: automatically generated dialogues based on either self-talk or bot-bot talk as a viable basis to conduct the evaluations. Using automatically generated dialogues is preferable to the static-context setting as it captures the bot’s multi-turn behavior. In *Spot The Bot* we showed that using automatically generated dialogues, which are rated on their capabilities of behaving like humans, results in robust and reproducible results. For *AutoJudge* we showed that self-talk conversations, which are annotated on the turn level, can be used as a basis to train a regression model, which achieves comparable correlations to human judgments as those trained on the static-context setting (such as ADEM or RUBER).

However, there are still open problems that need to be addressed to advance the field of evaluation of dialogue systems. The problems can be categorized into two sub-problems - the difficulty of measuring the quality of dialogues via human evaluation in a robust manner and the problem of not having reliable automated metrics.

**Human Evaluation.** Currently, there is no protocol for conducting a human evaluation, be it for creating human-to-bot conversation or for letting humans annotate dialogues or turns with regards to various features. The work by Finch and Choi [2020] investigated the human evaluation protocols used in various publications and concluded that there is a high diversity in how the various authors of the papers define the protocols. For instance, there are three additional terms used for the concept of fluency (grammaticality, readability, and consistency), and in each paper a different wording is used to define the attribute they are evaluating.

Another problem in human evaluation is low agreement scores. Usually, labels with a low agreement are discarded (e.g., Lowe et al. [2017a] discarded all non-agreeing samples). The main cause for this phenomenon is tied to the previous problem. There is no clear definition of good dialogue or a good utterance. Different features hint at a higher quality (e.g., fluency, logical coherence, correct co-reference, etc.). However, asking for each feature decreases the efficiency of the evaluation and increases the cost. Unfortunately, the agreement usually decreases if an evaluation asks about the "overall quality" or "appropriateness" of an utterance because humans have

different interpretations of "overall quality.". Liang et al. [2020] propose to avoid using Likert scale-based evaluation, which leads to low agreements. Instead, they propose the usage of preference-based evaluations where the crowdworker is shown two outputs by two different systems, which achieves higher agreement scores.

*Spot The Bot* is a protocol that is robust regarding the problem of disagreement. This is due to the fact that *Spot The Bot* measures the human-likeness of utterances. However, it must be investigated how well the *Spot The Bot* setting is transferable to other settings. For instance, if it is applied to evaluating Grice's maxims or if we want to evaluate more fine-grained features of the bot, such as its ability to maintain a consistent personality. This needs to be further assessed.

**Automated Evaluation.** The current approaches to automating the evaluation procedures are based on so-called trained metrics. *AutoJudge* falls into this class of trained metrics. However, these metrics are far from being mature and ready-to-use. Our experiments showed how unstable these metrics are, making them easy to be fooled. This is also confirmed by the experiments conducted by Sai et al. [2019], which showed how small perturbations of the context drastically changed the scores of trained metrics.

One major issue with the current research of trained metrics is the evaluation of those metrics themselves. Currently, they are all evaluated on how well they correlate to human judgments. However, it is unclear what this means exactly when the metric should be used as an evaluation tool for a dialogue system. A stronger evaluation of trained metrics would evaluate their ability to recreate a ranking, which resembles a ranking created by humans. In a small scale experiment, we tested the capability of RUBER to create a significant ranking. For this, we trained RUBER on PersonaChat data in the static-context setting. We let the 7 bots from the *Spot The Bot* setting create an answer-utterance for each static-context, which RUBER rated. For comparison, we annotated 100 static-contexts alongside the 7 utterances on a Likert-scale of 0 (bad), 1 (borderline), and 2 (good). Thus, there are 700 annotations. We generated rankings by pairwise comparisons. More precisely, for each static-context, we evaluated each pair of bots. The winner of a context is defined by the bot that gets the highest score. When comparing the ranking achieved by comparing RUBER scores and human scores, the problem becomes evident. Whereas the human-based evaluation achieves a sensible ranking, the RUBER based ranking is not significant, and it puts the retrieval based bots over the generative bots. However, the Spearman correlation between the RUBER scores and the human scores lies at 0.39, which is in line with the numbers reported by the authors. Thus, the evaluation of trained metrics themselves does not accurately measure how reliable

the metric is.

Another fundamental issue with trained metrics is the fact that these metrics resemble dialogue systems themselves. In fact, RUBER, ADEM, and AutoJudge can be interpreted as retrieval based dialogue systems as they score an utterance for a given context. The USR metric, on the other hand, is a mixture of a retrieval-based dialogue system and a generative dialogue system. Thus, it is not clear what kinds of biases are introduced in the evaluation. For instance, as stated above, RUBER rated retrieval based systems higher than generative systems, which might stem from the fact that RUBER is essentially a retrieval-based system itself.

To advance the field of dialogue systems in general, the development of automated metrics is important. However, current automated metrics are not yet applicable - the problem of evaluating the methods themselves needs to be solved. A first step would require to understand the capabilities of current automated metrics further. For instance, a benchmark that measures the evaluation method's capabilities to evaluate various features correctly would reveal where the methods still are lacking. Furthermore, we propose to change the goal of getting a high correlation to human judgments to recreating a ranking, which is a more concrete goal. As of yet is unclear how strong the relation between a dialogue system and its automated evaluation is. Since an evaluation method could be used as a retrieval-based dialogue system as well, for this reason, it is possible that a fully automated method is not feasible.

## 5.2 NLIDB

For NLIDB systems, we showed that unreferenced metrics are valid proxies for evaluation and can be used to improve the system. Access to unreferenced metrics opens up a variety of possibilities for automated or semi-automated improvements of NLIDB systems. We presented a post-processing-based application where the method is used to re-rank a set of the beam search hypothesis. Related to this application is using the unreferenced method as a meta-selection module, i.e., to select a response from different system's outputs.

However, there are more exciting applications of unreferenced metrics to improve a system via interaction.

**Active Learning.** Since the unreferenced metric can find wrong samples, it can be applied in the active learning setting to select annotation samples. There are some open questions regarding the feasibility of this.

- **Sample Selection.** It is unclear if those samples are found, which lead to the most improvement. Since deep-learning algorithms do not learn the same way as humans, the selection process will probably not be straightforward.
- **Noise.** Since the unreferenced metric is not perfect, i.e., it makes errors in both directions, making it unclear how much the noise impacts the process.
- **Budget.** Not all the samples should be annotated otherwise, the costs will be high at the beginning. Thus, it is important to determine how much should be annotated.
- **Improvement.** There are different ways to improve the system. The most trivial is to add the samples to the dataset. However, there are more possibilities to exploit the information. For instance, incorrect samples could be exploited as negative samples during the training procedure.

**Interactive Semantic Parsing.** A different application is in interactive semantic parsing, which extends the scenario to a multi-turn setting. In this case, the interaction with the human can be leveraged to correct the mistakes made. There are different ways on how to model the interaction. In section 2.5.3, the strategy of using leveraging free-form feedback to repeat the parsing process is presented. However, we can envision strategies that leverage the structured nature of the task. For instance, a more fine-grained analysis could reveal the type of error made by the parser and ask for a specific intervention by the user.

**Query Construction.** Our results show that extra hard questions are rarely solved correctly. A closer look reveals that these questions are usually very long. For instance, a query that contains three *Filter* operations and a *GroupBy* operation usually is associated with a very long question, which usually no human would ask. A solution would be to construct the query iteratively by a series of instructions. The tree structure would support this kind of interaction. For instance, the constraints could be inserted one after the other. Moreover, adding a *GroupBy* operation could also be stated in one command. In this scenario, the unreferenced method would monitor the progress of the query construction.

## 5.3 Transfer to another Language

The work presented in this thesis focuses only on systems in English. In this section, we sketch how these insights can be applied to other languages.

### 5.3.1 Conversational Dialogue Systems

*Spot The Bot* is language-agnostic since it depends on the dialogue systems under evaluation. The same setting can be applied to a pool of German-speaking dialogue systems with no change to the protocol. More interesting is the case of *AutoJudge*, where a model is built that automatically rates an utterance for a given context. Here, the neural network needs to be adapted to handle the target language. For languages with ample resources, this is a question of using a different pre-processing pipeline, using different word embeddings, or a different language model. However, for low-resource language, the situation is different. There the creation of word embeddings or large-scale language models is not feasible. Thus, different approaches must be applied, e.g., a bag-of-words classifier with manually created features. Thus, it is not clear how well the approach translates to these scenarios.

### 5.3.2 NLIDB

For the NLIDB systems, multiple steps need to be implemented to adapt the approaches to a different language.

**Data Collection.** There are two challenges here. First, the contents of the database must be adapted to the new language (if that is not already given). Assume that we want to adapt the *Moviedata* to Italian. First, we need to update all the movie titles to the Italian title, the movie descriptions, names of countries, and cities, among others. To apply the inverted annotation procedure, we need to recruit workers that know the target language and have basic SQL skills. For low-resource languages, there might not be many available workers that fit these criteria.

**NLIDB Development.** For training the NLIDB system, we face similar challenges to the ones we faced for *AutoJudge*. We need to update the pre-processing pipeline, the word embeddings, or the language model. Note that this is only needed for the encoder, as the decoder only depends on the operation tree rules. However, if we operate in a low-resource regime, and a sophisticated neural encoder is not an option, a different method of encoding the question must be devised. Note that we cannot simply build a feature-based classifier since we need to generate an OT in this case. Thus, we need to either use a more straightforward encoding (e.g., bad-of-words) and feed this encoding directly into the decoder or apply a completely different approach. For instance, we can apply a classical pattern-based approach.

**Back-translation.** To back-translate the OT to a natural language question, we need to adapt the patterns that the domain expert manually defines. That is, the

domain expert needs to translate the general production rules and the domain-specific rules manually.

**Semantic Similarity.** The semantic similarity module needs to be specifically trained for the target language. In this work, we used the out-of-the-box version of NUBIA, which only works for English. For languages with ample resources, we have to adapt the pre-processing pipeline, using different word embeddings or a different language model. Then we need data to train or fine-tune the semantic similarity model, which needs to be collected for some languages. For low-resource settings, again, we need to opt for different models, which are less data-intensive.

Adapting a model to a different language is no trivial task. In some cases, the adaption fails due to insufficient data. Especially, the adaption of the work on NLIDB requires many components to be translated. This complexity might lead to the system not working as well as it does for English.

# References

- D. Adiwardana, M.-T. Luong, D. R. So, J. Hall, N. Fiedel, R. Thoppilan, Z. Yang, A. Kulshreshtha, G. Nemade, Y. Lu, et al. Towards a Human-like Open-Domain Chatbot. *arXiv preprint arXiv:2001.09977*, 2020.
- K. Affolter, K. Stockinger, and A. Bernstein. A comparative survey of recent natural language interfaces for databases. *The VLDB Journal*, 28(5):793–819, Oct 2019. ISSN 0949-877X. doi: 10.1007/s00778-019-00567-8. URL <https://doi.org/10.1007/s00778-019-00567-8>.
- E. Agirre, C. Banea, D. Cer, M. Diab, A. Gonzalez-Agirre, R. Mihalcea, G. Rigau, and J. Wiebe. SemEval-2016 Task 1: Semantic Textual Similarity, Monolingual and Cross-Lingual Evaluation. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 497–511, San Diego, California, June 2016. Association for Computational Linguistics. doi: 10.18653/v1/S16-1081. URL <https://www.aclweb.org/anthology/S16-1081>.
- J. Amidei, P. Piwek, and A. Willis. The use of rating and Likert scales in Natural Language Generation human evaluation tasks: A review and some recommendations. In *Proceedings of the 12th International Conference on Natural Language Generation*, pages 397–402, Tokyo, Japan, November 2019a. Association for Computational Linguistics.
- J. Amidei, P. Piwek, and A. Willis. Agreement is overrated: A plea for correlation to assess human evaluation reliability. In *Proceedings of the 12th International Conference on Natural Language Generation*, pages 344–354, 2019b.
- C. Anderson-Bergman. icenReg: Regression Models for Interval Censored Data in R. *Journal of Statistical Software, Articles*, 81(12):1–23, 2017. ISSN 1548-7660. doi: 10.18637/jss.v081.i12. URL <https://www.jstatsoft.org/v081/i12>.
- F. Basik, B. Hättasch, A. Ilkhechi, A. Usta, S. Ramaswamy, P. Utama, N. Weir, C. Binnig, and U. Cetintemel. DBPal: A Learned NL-Interface for Databases. SIGMOD ’18, page 1765–1768, New York, NY, USA, 2018. Association for

- Computing Machinery. ISBN 9781450347037. doi: 10.1145/3183713.3193562. URL <https://doi.org/10.1145/3183713.3193562>.
- H. Bast and E. Haussmann. More Accurate Question Answering on Freebase. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pages 1431–1440. ACM, 2015.
- P. A. Bishop and R. L. Herron. Use and Misuse of the Likert Item Responses and Other Ordinal Measures. *International journal of exercise science*, 8(3):297, 2015.
- L. Blunschi, C. Jossen, D. Kossmann, M. Mori, and K. Stockinger. SODA: Generating SQL for Business Users. *Proceedings of the VLDB Endowment*, 5(10):932–943, 2012.
- P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov. Enriching Word Vectors with Subword Information. *Transactions of the Association for Computational Linguistics*, 5:135–146, 2017. doi: 10.1162/tacl\_a\_00051. URL <https://www.aclweb.org/anthology/Q17-1010>.
- O. Bojar, C. Buck, C. Callison-Burch, C. Federmann, B. Haddow, P. Koehn, C. Monz, M. Post, R. Soricut, and L. Specia. Findings of the 2013 Workshop on Statistical Machine Translation. In *Proceedings of the Eighth Workshop on Statistical Machine Translation*, pages 1–44, Sofia, Bulgaria, Aug. 2013. Association for Computational Linguistics. URL <https://www.aclweb.org/anthology/W13-2201>.
- E. Bruni and R. Fernandez. Adversarial Evaluation for Open-Domain Dialogue Generation. In *Proceedings of the SIGDIAL 2017 Conference: The 18th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 284–288. Association for Computational Linguistics, 2017.
- J. A. Campos, A. Otegi, A. Soroa, J. Deriu, M. Cieliebak, and E. Agirre. DoQA - Accessing Domain-Specific FAQs via Conversational QA. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7302–7314, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.652. URL <https://www.aclweb.org/anthology/2020.acl-main.652>.
- H. Chen, X. Liu, D. Yin, and J. Tang. A Survey on Dialogue Systems: Recent Advances and New Frontiers. *Special Interest Group on Knowledge Discovery and Data Mining (SIGKDD) Explor. Newsl.*, 19(2):25–35, Nov. 2017. ISSN 1931-0145.



- J. Cheng, S. Reddy, V. Saraswat, and M. Lapata. Learning an Executable Neural Semantic Parser. *Computational Linguistics*, 45(1):59–94, Mar. 2019. doi: 10.1162/coli.a.00342. URL <https://www.aclweb.org/anthology/J19-1002>.
- E. Choi, H. He, M. Iyyer, M. Yatskar, W.-t. Yih, Y. Choi, P. Liang, and L. Zettlemoyer. QuAC: Question Answering in Context. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2174–2184, Brussels, Belgium, Oct.-Nov. 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1241. URL <https://www.aclweb.org/anthology/D18-1241>.
- J. Chung, C. Gulcehre, K. Cho, and Y. Bengio. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. *arXiv preprint arXiv:1412.3555*, 2014.
- K. M. Colby. Modeling a paranoid mind. *Behavioral and Brain Sciences*, 4(4): 515–534, 1981.
- M. A. Covington and J. D. McFall. Cutting the Gordian Knot: The Moving-Average Type-Token Ratio (MATTR). *Journal of Quantitative Linguistics*, 17(2):94–100, 2010. doi: 10.1080/09296171003643098.
- D. R. Cox. Regression Models and Life-Tables. *Journal of the Royal Statistical Society. Series B (Methodological)*, 34(2):187–220, 1972. ISSN 00359246. URL <http://www.jstor.org/stable/2985181>.
- J. Deriu and M. Cieliebak. Towards a Metric for Automated Conversational Dialogue System Evaluation and Improvement. In *Proceedings of the 12th International Conference on Natural Language Generation*, pages 432–437, Tokyo, Japan, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/W19-8654. URL <https://www.aclweb.org/anthology/W19-8654>.
- J. Deriu, K. Mlynchik, P. Schläpfer, A. Rodrigo, D. von Grünigen, N. Kaiser, K. Stockinger, E. Agirre, and M. Cieliebak. A Methodology for Creating Question Answering Corpora Using Inverse Data Annotation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 897–911, Online, July 2020a. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.84. URL <https://www.aclweb.org/anthology/2020.acl-main.84>.
- J. Deriu, A. Rodrigo, A. Otegi, G. Echegoyen, S. Rosset, E. Agirre, and M. Cieliebak. Survey on Evaluation Methods for Dialogue Systems. *Artificial Intelligence Review*, pages 1–56, 2020b.

- J. Deriu, D. Tuggener, P. von Däniken, J. A. Campos, A. Rodrigo, T. Belkacem, A. Soroa, E. Agirre, and M. Cieliebak. Spot The Bot: A Robust and Efficient Framework for the Evaluation of Conversational Dialogue Systems. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3971–3984, Online, Nov. 2020c. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.326. URL <https://www.aclweb.org/anthology/2020.emnlp-main.326>.
- J. M. Deriu and M. Cieliebak. SwissAlps at SemEval-2017 Task 3: Attention-based Convolutional Neural Network for Community Question Answering. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 334–338, Vancouver, Canada, Aug. 2017. Association for Computational Linguistics. doi: 10.18653/v1/S17-2054. URL <https://www.aclweb.org/anthology/S17-2054>.
- J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL <https://www.aclweb.org/anthology/N19-1423>.
- E. Dinan, S. Roller, K. Shuster, A. Fan, M. Auli, and J. Weston. Wizard of Wikipedia: Knowledge-Powered Conversational Agents. In *International Conference on Learning Representations*, 2019. URL <https://openreview.net/forum?id=r1l73iRqKm>.
- E. Dinan, V. Logacheva, V. Malykh, A. Miller, K. Shuster, J. Urbanek, D. Kiela, A. Szlam, I. Serban, R. Lowe, S. Prabhumoye, A. W. Black, A. Rudnicky, J. Williams, J. Pineau, M. Burtsev, and J. Weston. The Second Conversational Intelligence Challenge (ConvAI2). In S. Escalera and R. Herbrich, editors, *The NeurIPS ’18 Competition*, pages 187–208, Cham, 2020. Springer International Publishing. ISBN 978-3-030-29135-8.
- M. Dubey, D. Banerjee, A. Abdelkawi, and J. Lehmann. LC-QuAD 2.0: A Large Dataset for Complex Question Answering over Wikidata and DBpedia. In C. Ghidini, O. Hartig, M. Maleshkova, V. Svátek, I. Cruz, A. Hogan, J. Song, M. Lefrançois, and F. Gandon, editors, *The Semantic Web – ISWC 2019*, pages 69–78, Cham, 2019. Springer International Publishing. ISBN 978-3-030-30796-7.

- O. Dušek, J. Novikova, and V. Rieser. Findings of the E2E NLG Challenge. In *Proceedings of the 11th International Conference on Natural Language Generation*, pages 322–328, Tilburg University, The Netherlands, Nov. 2018. Association for Computational Linguistics. doi: 10.18653/v1/W18-6539. URL <https://www.aclweb.org/anthology/W18-6539>.
- A. Elgohary, S. Hosseini, and A. Hassan Awadallah. Speak to your Parser: Interactive Text-to-SQL with Natural Language Feedback. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2065–2077, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.187. URL <https://www.aclweb.org/anthology/2020.acl-main.187>.
- K. Evanini, P. Hunter, J. Liscombe, D. Suendermann, K. Dayanidhi, and R. Pieraccini. Caller Experience: A method for evaluating dialog systems and its automatic prediction. In *2008 IEEE Spoken Language Technology Workshop*, pages 129–132, Goa, India, Dec 2008. doi: 10.1109/SLT.2008.4777857.
- A. Eyal, L. Rokach, M. Kalech, O. Amir, R. Chougule, R. Vaidyanathan, and K. Pattada. Survival Analysis of Automobile Components Using Mutually Exclusive Forests. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 44(2):246–253, 2014.
- S. E. Finch and J. D. Choi. Towards Unified Dialogue System Evaluation: A Comprehensive Analysis of Current Evaluation Protocols. In *Proceedings of the 21th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 236–245, 1st virtual meeting, July 2020. Association for Computational Linguistics. URL <https://www.aclweb.org/anthology/2020.sigdial-1.29>.
- C. Finegan-Dollak, J. K. Kummerfeld, L. Zhang, K. Ramanathan, S. Sadasivam, R. Zhang, and D. Radev. Improving Text-to-SQL Evaluation Methodology. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 351–360, Melbourne, Australia, July 2018. Association for Computational Linguistics. doi: 10.18653/v1/P18-1033. URL <https://www.aclweb.org/anthology/P18-1033>.
- J. L. Fleiss. Measuring nominal scale agreement among many raters. *Psychological bulletin*, 76(5):378–382, 1971. doi: <http://dx.doi.org/10.1037/h0031619>.
- S. Gandhe and D. Traum. A Semi-automated Evaluation Metric for Dialogue Model Coherence. In A. Rudnicky, A. Raux, I. Lane, and T. Misu, editors, *Situated Dialog in Speech-Based Human-Computer Interaction*, pages 217–225.

- Springer International Publishing, Cham, 2016. ISBN 978-3-319-21834-2. doi: 10.1007/978-3-319-21834-2\_19. URL [https://doi.org/10.1007/978-3-319-21834-2\\_19](https://doi.org/10.1007/978-3-319-21834-2_19).
- J. Gao, M. Galley, and L. Li. Neural Approaches to Conversational AI. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, SIGIR '18, page 1371–1374, New York, NY, USA, 2018. Association for Computing Machinery. ISBN 9781450356572. doi: 10.1145/3209978.3210183. URL <https://doi.org/10.1145/3209978.3210183>.
- M. Gašić, F. Jurčićek, B. Thomson, K. Yu, and S. Young. On-line policy optimisation of spoken dialogue systems via live interaction with human subjects. In *2011 IEEE Workshop on Automatic Speech Recognition Understanding*, pages 312–317, Dec 2011. doi: 10.1109/ASRU.2011.6163950.
- A. Ghandeharioun, J. H. Shen, N. Jaques, C. Ferguson, N. Jones, A. Lapedriza, and R. Picard. Approximating Interactive Human Evaluation with Self-Play for Open-Domain Dialog Systems. In *Advances in Neural Information Processing Systems*, pages 13658–13669, 2019.
- I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative Adversarial Nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, NIPS 27, pages 2672–2680. Curran Associates, Inc., 2014. URL <http://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf>.
- H. P. Grice. Logic and Conversation. In *Speech acts*, pages 41–58. Brill, 1975.
- J. Guo, Z. Zhan, Y. Gao, Y. Xiao, J.-G. Lou, T. Liu, and D. Zhang. Towards Complex Text-to-SQL in Cross-Domain Database with Intermediate Representation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4524–4535, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1444. URL <https://www.aclweb.org/anthology/P19-1444>.
- C. T. Hemphill, J. J. Godfrey, and G. R. Doddington. The ATIS Spoken Language Systems Pilot Corpus. In *Speech and Natural Language: Proceedings of a Workshop Held at Hidden Valley, Pennsylvania, June 24-27, 1990*, 1990. URL <https://www.aclweb.org/anthology/H90-1021>.
- R. Herbrich, T. Minka, and T. Graepel. TrueSkillTM: A Bayesian Skill Rating System. In *Proceedings of the 19th International Conference on Neural*

- Information Processing Systems*, NIPS'06, page 569–576, Cambridge, MA, USA, 2006. MIT Press.
- S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, pages 1735–1780, 1997.
- D. Hovy, T. Berg-Kirkpatrick, A. Vaswani, and E. Hovy. Learning Whom to Trust with MACE. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1120–1130, Atlanta, Georgia, June 2013. Association for Computational Linguistics. URL <https://www.aclweb.org/anthology/N13-1132>.
- S. Iyer, I. Konstas, A. Cheung, J. Krishnamurthy, and L. Zettlemoyer. Learning a Neural Semantic Parser from User Feedback. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 963–973, Vancouver, Canada, July 2017. Association for Computational Linguistics. doi: 10.18653/v1/P17-1089. URL <https://www.aclweb.org/anthology/P17-1089>.
- D. Jurafsky and J. H. Martin. *Speech and Language Processing*, chapter Dialog Systems and Chatbots. Draft of 3rd edition, 2017.
- H. Kane, M. Y. Kocyigit, A. Abdalla, P. Ajanoh, and M. Coulibali. NUBIA: Neural Based Interchangeability Assessor for Text Generation. *arXiv preprint arXiv:2004.14667*, 2020.
- A. Kannan and O. Vinyals. Adversarial Evaluation of Dialogue Models. *Workshop on Adversarial Training at Neural Information Processing Systems 2016*, 2016.
- I. Labutov, B. Yang, and T. Mitchell. Learning to Learn Semantic Parsers from Natural Language Supervision. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1676–1690, Brussels, Belgium, Oct.-Nov. 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1195. URL <https://www.aclweb.org/anthology/D18-1195>.
- C. Lee, S. Jung, S. Kim, and G. G. Lee. Example-based dialog modeling for practical multi-domain dialog system. *Speech Communication*, 51(5):466–484, 2009.
- J. Lehmann, R. Isele, M. Jakob, A. Jentzsch, D. Kontokostas, P. N. Mendes, S. Hellmann, M. Morsey, P. van Kleef, S. Auer, and C. Bizer. DBpedia - A Large-scale, Multilingual Knowledge Base Extracted from Wikipedia. *Semantic*

- Web*, 6(2):167–195, 2015. doi: 10.3233/SW-140134. URL <https://madoc.bib.uni-mannheim.de/37476/>.
- F. Li and H. Jagadish. Constructing an Interactive Natural Language Interface for Relational Databases. *Proceedings of the VLDB Endowment*, 8(1):73–84, 2014.
- J. Li and S. Ma. *Survival Analysis in Medicine and Genetics*. CRC Press, 2013.
- J. Li, M. Galley, C. Brockett, J. Gao, and B. Dolan. A Diversity-Promoting Objective Function for Neural Conversation Models. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 110–119, San Diego, California, 2016a. Association for Computational Linguistics. URL <http://www.aclweb.org/anthology/N16-1014>.
- J. Li, W. Monroe, A. Ritter, D. Jurafsky, M. Galley, and J. Gao. Deep Reinforcement Learning for Dialogue Generation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1192–1202, Austin, Texas, Nov. 2016b. Association for Computational Linguistics. doi: 10.18653/v1/D16-1127. URL <https://www.aclweb.org/anthology/D16-1127>.
- Y. Li, H. Su, X. Shen, W. Li, Z. Cao, and S. Niu. DailyDialog: A Manually Labelled Multi-turn Dialogue Dataset. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 986–995, Taipei, Taiwan, Nov. 2017. Asian Federation of Natural Language Processing. URL <https://www.aclweb.org/anthology/I17-1099>.
- W. Liang, J. Zou, and Z. Yu. Beyond User Self-Reported Likert Scale Ratings: A Comparison Model for Automatic Dialog Evaluation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1363–1374, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.126. URL <https://www.aclweb.org/anthology/2020.acl-main.126>.
- C.-Y. Lin. ROUGE: A Package for Automatic Evaluation of Summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain, July 2004. Association for Computational Linguistics. URL <https://www.aclweb.org/anthology/W04-1013>.
- C.-W. Liu, R. Lowe, I. Serban, M. Noseworthy, L. Charlin, and J. Pineau. How NOT To Evaluate Your Dialogue System: An Empirical Study of Unsupervised

- Evaluation Metrics for Dialogue Response Generation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2122–2132, Austin, Texas, Nov. 2016. Association for Computational Linguistics. doi: 10.18653/v1/D16-1230. URL <https://www.aclweb.org/anthology/D16-1230>.
- H. Liu, L. Fang, Q. Liu, B. Chen, J.-G. Lou, and Z. Li. Leveraging Adjective-Noun Phrasing Knowledge for Comparison Relation Prediction in Text-to-SQL. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3515–3520, Hong Kong, China, Nov. 2019a. Association for Computational Linguistics. doi: 10.18653/v1/D19-1356. URL <https://www.aclweb.org/anthology/D19-1356>.
- Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *arXiv preprint arXiv:1907.11692*, 2019b.
- R. Lowe, N. Pow, I. Serban, and J. Pineau. The Ubuntu Dialogue Corpus: A Large Dataset for Research in Unstructured Multi-Turn Dialogue Systems. In *Proceedings of the SIGDIAL 2015 Conference: The 16th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 285–294, Prague, Czech Republic, September 2015. Association for Computational Linguistics. URL <http://aclweb.org/anthology/W15-4640>.
- R. Lowe, I. V. Serban, M. Noseworthy, L. Charlin, and J. Pineau. On the Evaluation of Dialogue Systems with Next Utterance Classification. In *Proceedings of the SIGDIAL 2016 Conference: The 17th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 264–269, Los Angeles, CA, USA, 2016. Association for Computational Linguistics. URL <http://www.aclweb.org/anthology/W16-3634>.
- R. Lowe, M. Noseworthy, I. V. Serban, N. Angelard-Gontier, Y. Bengio, and J. Pineau. Towards an Automatic Turing Test: Learning to Evaluate Dialogue Responses. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1116–1126, Vancouver, Canada, July 2017a. Association for Computational Linguistics. doi: 10.18653/v1/P17-1103. URL <https://www.aclweb.org/anthology/P17-1103>.
- R. Lowe, N. Pow, I. V. Serban, L. Charlin, C.-W. Liu, and J. Pineau. Training End-to-End Dialogue Systems with the Ubuntu Dialogue Corpus. *Dialogue &*

- Discourse*, 8(1):31–65, 2017b. URL <http://dad.uni-bielefeld.de/index.php/dad/article/view/3698>.
- T. Luong, H. Pham, and C. D. Manning. Effective Approaches to Attention-based Neural Machine Translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, Lisbon, Portugal, Sept. 2015. Association for Computational Linguistics. doi: 10.18653/v1/D15-1166. URL <https://www.aclweb.org/anthology/D15-1166>.
- S. Mehri and M. Eskenazi. USR: An Unsupervised and Reference Free Evaluation Metric for Dialog Generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 681–707, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.64. URL <https://www.aclweb.org/anthology/2020.acl-main.64>.
- A. Miller, W. Feng, D. Batra, A. Bordes, A. Fisch, J. Lu, D. Parikh, and J. Weston. ParlAI: A Dialog Research Software Platform. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, EMNLP ’17, pages 79–84, 2017. URL <https://www.aclweb.org/anthology/D17-2014>.
- T. Paek. Empirical Methods for Evaluating Dialog Systems. In *Proceedings of the Second SIGdial Workshop on Discourse and Dialogue - Volume 16*, SIGDIAL ’01, pages 1–9, Stroudsburg, PA, USA, 2001. Association for Computational Linguistics. URL <https://doi.org/10.3115/1118078.1118092>.
- K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu. Bleu: a Method for Automatic Evaluation of Machine Translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA, July 2002. Association for Computational Linguistics. doi: 10.3115/1073083.1073135. URL <https://www.aclweb.org/anthology/P02-1040>.
- A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever. Language Models are Unsupervised Multitask Learners. *Technical report, OpenAI*, 2019.
- H. Rashkin, E. M. Smith, M. Li, and Y.-L. Boureau. Towards Empathetic Open-domain Conversation Models: A New Benchmark and Dataset. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5370–5381, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1534. URL <https://www.aclweb.org/anthology/P19-1534>.



- S. Reddy, D. Chen, and C. D. Manning. CoQA: A Conversational Question Answering Challenge. *Transactions of the Association for Computational Linguistics*, 7:249–266, 2018. URL <https://www.aclweb.org/anthology/Q19-1016>.
- A. Ritter, C. Cherry, and B. Dolan. Unsupervised Modeling of Twitter Conversations. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 172–180, Los Angeles, California, June 2010. Association for Computational Linguistics. URL <https://www.aclweb.org/anthology/N10-1020>.
- A. Ritter, C. Cherry, and W. B. Dolan. Data-driven Response Generation in Social Media. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing, EMNLP '11*, pages 583–593, Edinburgh, Scotland, UK., 2011. ISBN 978-1-937284-11-4. URL <http://dl.acm.org/citation.cfm?id=2145432.2145500>.
- S. Roller, E. Dinan, N. Goyal, D. Ju, M. Williamson, Y. Liu, J. Xu, M. Ott, K. Shuster, E. M. Smith, et al. Recipes for building an open-domain chatbot. *arXiv preprint arXiv:2004.13637*, 2020.
- D. Saha, A. Floratou, K. Sankaranarayanan, U. F. Minhas, A. R. Mittal, and F. Özcan. ATHENA: An Ontology-Driven System for Natural Language Querying over Relational Data Stores. *Proceedings of the VLDB Endowment*, 9(12):1209–1220, 2016.
- A. B. Sai, M. D. Gupta, M. M. Khapra, and M. Srinivasan. Re-Evaluating ADEM: A Deeper Look at Scoring Dialogue Responses. In *Proceedings of the thirty-third AAAI Conference on Artificial Intelligence*, volume 33 of *AAAI'19*, pages 6220–6227, Honolulu, Hawaii, USA, 2019. URL <https://aaai.org/ojs/index.php/AAAI/article/view/4581>.
- J. Schatzmann, B. Thomson, K. Weilhammer, H. Ye, and S. Young. Agenda-based User Simulation for Bootstrapping a POMDP Dialogue System. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Companion Volume, Short Papers*, NAACL-Short '07, pages 149–152, Rochester, New York, 2007. URL <http://dl.acm.org/citation.cfm?id=1614108.1614146>.
- A. Schmitt and S. Ultes. Interaction Quality: Assessing the quality of ongoing spoken dialog interaction by experts—And how it relates to user satisfaction.

- Speech Communication*, 74:12–36, 2015. ISSN 0167-6393. URL <https://dl.acm.org/doi/10.1016/j.specom.2015.06.003>.
- I. V. Serban, A. Sordoni, Y. Bengio, A. Courville, and J. Pineau. Building End-to-end Dialogue Systems Using Generative Hierarchical Neural Network Models. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI’16, pages 3776–3783, Phoenix, Arizona, USA, 2016. AAAI Press. URL <http://dl.acm.org/citation.cfm?id=3016387.3016435>.
- I. V. Serban, T. Klinger, G. Tesauro, K. Talamadupula, B. Zhou, Y. Bengio, and A. C. Courville. Multiresolution Recurrent Neural Networks: An Application to Dialogue Response Generation. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, AAAI ’17, pages 3288–3294, San Francisco, California, USA, 2017a. URL <http://aaai.org/ocs/index.php/AAAI/AAAI17/paper/view/14571>.
- I. V. Serban, C. Sankar, M. Germain, S. Zhang, Z. Lin, S. Subramanian, T. Kim, M. Pieper, S. Chandar, N. R. Ke, et al. A Deep Reinforcement Learning Chatbot. *arXiv preprint arXiv:1709.02349*, 2017b.
- I. V. Serban, A. Sordoni, R. Lowe, L. Charlin, J. Pineau, A. Courville, and Y. Bengio. A Hierarchical Latent Variable Encoder-Decoder Model for Generating Dialogues. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, AAAI’17, pages 3295–3301, San Francisco, California USA, 2017c. URL <https://dl.acm.org/doi/10.5555/3298023.3298047>.
- I. V. Serban, R. Lowe, P. Henderson, L. Charlin, and J. Pineau. A Survey of Available Corpora for Building Data-Driven Dialogue Systems: The Journal Version. *Dialogue & Discourse*, 1(9), 2018.
- L. Shang, Z. Lu, and H. Li. Neural Responding Machine for Short-Text Conversation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, ACL - IJCNLP ’15, pages 1577–1586, Beijing, China, 2015. URL <http://www.aclweb.org/anthology/P15-1152>.
- A. Simitsis, G. Koutrika, and Y. Ioannidis. Précis: from unstructured keywords as queries to structured databases as answers. *The VLDB Journal—The International Journal on Very Large Data Bases*, 17(1):117–149, 2008.
- A. Sordoni, M. Galley, M. Auli, C. Brockett, Y. Ji, M. Mitchell, J.-Y. Nie, J. Gao, and B. Dolan. A Neural Network Approach to Context-Sensitive Generation of

- Conversational Responses. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 196–205, Denver, Colorado, May–June 2015. Association for Computational Linguistics. doi: 10.3115/v1/N15-1020. URL <https://www.aclweb.org/anthology/N15-1020>.
- Z. Sun, Q. Zhu, L. Mou, Y. Xiong, G. Li, and L. Zhang. A Grammar-Based Structural CNN Decoder for Code Generation. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):7055–7062, Jul. 2019. doi: 10.1609/aaai.v33i01.33017055. URL <https://ojs.aaai.org/index.php/AAAI/article/view/4686>.
- I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to Sequence Learning with Neural Networks. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS’14, pages 3104–3112, Cambridge, MA, USA, 2014. MIT Press. URL <http://dl.acm.org/citation.cfm?id=2969033.2969173>.
- R. S. Sutton and A. G. Barto. *Policy Gradient Methods*, chapter 13. MIT press, 2 edition, 2018.
- L. R. Tang and R. J. Mooney. Automated Construction of Database Interfaces: Intergrating Statistical and Relational Learning for Semantic Parsing. In *2000 Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, pages 133–141, Hong Kong, China, Oct. 2000. Association for Computational Linguistics. doi: 10.3115/1117794.1117811. URL <https://www.aclweb.org/anthology/W00-1317>.
- C. Tao, L. Mou, D. Zhao, and R. Yan. RUBER: An Unsupervised Method for Automatic Evaluation of Open-Domain Dialog Systems, 2018. URL <https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/16179/15752>.
- D. Traum, A. Jones, K. Hays, H. Maio, O. Alexander, R. Artstein, P. Debevec, A. Gainer, K. Georgila, K. Haase, K. Jungblut, A. Leuski, S. Smith, and W. Swartout. New Dimensions in Testimony: Digitally Preserving a Holocaust Survivor’s Interactive Storytelling. In H. Schoenau-Fog, L. E. Bruni, S. Louchart, and S. Baceviciute, editors, *Interactive Storytelling*, pages 269–281, Cham, 2015. Springer International Publishing. ISBN 978-3-319-27036-4.
- A. M. Turing. Computing Machinery and Intelligence. *Mind*, LIX(236):433–460, 1950. doi: 10.1093/mind/LIX.236.433. URL <http://dx.doi.org/10.1093/mind/LIX.236.433>.

- B. W. Turnbull. Nonparametric Estimation of a Survivorship Function with Doubly Censored Data. *Journal of the American Statistical Association*, 69 (345):169–173, 1974. doi: 10.1080/01621459.1974.10480146.
- O. Vinyals and Q. Le. A Neural Conversational Model. *arXiv preprint arXiv:1506.05869*, 2015.
- B. Wang, R. Shin, X. Liu, O. Polozov, and M. Richardson. RAT-SQL: Relation-Aware Schema Encoding and Linking for Text-to-SQL Parsers. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7567–7578, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.677. URL <https://www.aclweb.org/anthology/2020.acl-main.677>.
- Y. Wang, J. Berant, and P. Liang. Building a Semantic Parser Overnight. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1332–1342, Beijing, China, July 2015. Association for Computational Linguistics. doi: 10.3115/v1/P15-1129. URL <https://www.aclweb.org/anthology/P15-1129>.
- J. Weizenbaum. ELIZA - a Computer Program for the Study of Natural Language Communication Between Man and Machine. *Communications of the ACM*, 9(1): 36–45, Jan. 1966. ISSN 0001-0782. URL <http://doi.acm.org/10.1145/365153.365168>.
- T. Wolf, V. Sanh, J. Chaumond, and C. Delangue. TransferTransfo: A Transfer Learning Approach for Neural Network Based Conversational Agents. *arXiv preprint arXiv:1901.08149*, 2019.
- P. Yin and G. Neubig. A Syntactic Neural Model for General-Purpose Code Generation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 440–450, Vancouver, Canada, July 2017. Association for Computational Linguistics. doi: 10.18653/v1/P17-1041. URL <https://www.aclweb.org/anthology/P17-1041>.
- T. Yu, M. Yasunaga, K. Yang, R. Zhang, D. Wang, Z. Li, and D. Radev. SyntaxSQLNet: Syntax Tree Networks for Complex and Cross-Domain Text-to-SQL Task. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1653–1663, Brussels, Belgium, Oct.-Nov. 2018a. Association for Computational Linguistics. doi: 10.18653/v1/D18-1193. URL <https://www.aclweb.org/anthology/D18-1193>.

- T. Yu, R. Zhang, K. Yang, M. Yasunaga, D. Wang, Z. Li, J. Ma, I. Li, Q. Yao, S. Roman, Z. Zhang, and D. Radev. Spider: A Large-Scale Human-Labeled Dataset for Complex and Cross-Domain Semantic Parsing and Text-to-SQL Task. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3911–3921, Brussels, Belgium, Oct.-Nov. 2018b. Association for Computational Linguistics. doi: 10.18653/v1/D18-1425. URL <https://www.aclweb.org/anthology/D18-1425>.
- T. Yu, R. Zhang, M. Yasunaga, Y. C. Tan, X. V. Lin, S. Li, H. Er, I. Li, B. Pang, T. Chen, E. Ji, S. Dixit, D. Proctor, S. Shim, J. Kraft, V. Zhang, C. Xiong, R. Socher, and D. Radev. SPaC: Cross-Domain Semantic Parsing in Context. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4511–4523, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1443. URL <https://www.aclweb.org/anthology/P19-1443>.
- J. M. Zelle and R. J. Mooney. Learning to Parse Database Queries Using Inductive Logic Programming. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence - Volume 2, AAAI’96*, page 1050–1055. AAAI Press, 1996. ISBN 026251091X.
- S. Zhang, E. Dinan, J. Urbanek, A. Szlam, D. Kiela, and J. Weston. Personalizing Dialogue Agents: I have a dog, do you have pets too? In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2204–2213, Melbourne, Australia, July 2018. Association for Computational Linguistics. doi: 10.18653/v1/P18-1205. URL <https://www.aclweb.org/anthology/P18-1205>.
- Y. Zhang, S. Sun, M. Galley, Y.-C. Chen, C. Brockett, X. Gao, J. Gao, J. Liu, and B. Dolan. DIALOGPT : Large-Scale Generative Pre-training for Conversational Response Generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 270–278, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-demos.30. URL <https://www.aclweb.org/anthology/2020.acl-demos.30>.
- Q. Zhao and J. Sun. Generalized log-rank test for mixed interval-censored failure time data. *Statistics in Medicine*, 23(10):1621–1629, 2004. doi: 10.1002/sim.1746. URL <https://onlinelibrary.wiley.com/doi/abs/10.1002/sim.1746>.

- V. Zhong, C. Xiong, and R. Socher. Seq2SQL: Generating Structured Queries From Natural Language Using Reinforcement Learning, 2018. URL <https://openreview.net/forum?id=Syx6bz-Ab>.
- L. Zhou, J. Gao, D. Li, and H.-Y. Shum. The Design and Implementation of XiaoIce, an Empathetic Social Chatbot. *Computational Linguistics*, 46(1):53–93, 2020.